



OCPP 2.0.1

Part 0 - Introduction

FINAL, 2020-03-31

Table of Contents

Disclaimer	1
Version History	2
1. Introduction.....	3
1.1. OCPP version 2.0.1.....	3
1.2. Terms and abbreviations.....	3
1.3. References	4
2. New functionalities in OCPP2.0	5
2.1. Device Management	5
2.2. Improvements for better handling of large amounts of transactions	5
2.3. Improvements regarding cyber security	5
2.4. Extended Smart Charging	5
2.5. Support for ISO 15118	6
2.6. Improvements for customer experience	6
2.7. Transport Protocols: OCPP-J Improvements	6
2.8. Minor changes/extensions	7
3. OCPP 2.0 Documentation Structure.....	8
3.1. Overview of Specification Parts	8
3.2. Functional Blocks.....	9
3.3. All Functional Blocks and use cases	10
4. Basic implementation of OCPP 2.0	13

Disclaimer

Copyright © 2010 – 2020 Open Charge Alliance. All rights reserved.

This document is made available under the **Creative Commons Attribution-NoDerivatives 4.0 International Public License** (<https://creativecommons.org/licenses/by-nd/4.0/legalcode>).

Version History

Version	Date	Author	Description
2.0.1	2020-03-31	Franc Buve (OCA) Paul Klapwijk (OCA) Robert de Leeuw (<i>IHomer</i>)	Final version of OCPP 2.0.1
2.0	2018-04-11	Milan Jansen (OCA) Paul Klapwijk (OCA) Robert de Leeuw (<i>IHomer</i>) Robben Riksen (<i>Alliander</i>)	OCPP 2.0 April 2018 First release of this Introduction document

1. Introduction

Electric Vehicles (EVs) are becoming the new standard for mobility all over the world. This development is only possible with a good coverage of Charging Stations. To advance the roll out of charging infrastructure, open communication standards play a key role: to enable switching from charging network without necessarily replacing all the Charging Stations, to encourage innovation and cost effectiveness and to allow many and diverse players participate in this new industry.

Additionally, the EV charging infrastructure is part of the Smart Grid, a larger and still evolving ecosystem of actors, devices and protocols. In this Smart Grid ecosystem, open communications standards are key enablers for two-way power flows, real time information exchange, demand control and eMobility services.

The Open Charge Point Protocol (OCPP) is the industry-supported de facto standard for communication between a Charging Station and a Charging Station Management System (CSMS) and is designed to accommodate any type of charging technique. OCPP is an open standard with no cost or licensing barriers for adoption.

1.1. OCPP version 2.0.1

This specification defines version 2.0.1 of OCPP.

After the release of OCPP 2.0, some issues were found in OCPP 2.0. Some of these issues could not be fixed issuing errata to the specification text only, as has been done with OCPP 1.6, but required changes to the protocol's machine-readable schema definition files that cannot be backward compatible.

To prevent confusion in the market and possible interoperability issues in the field, OCA has decided to name this version: 2.0.1. OCPP 2.0.1 contains fixes for all the known issues, to date, not only the fixes to the messages.

This version replaces OCPP 2.0. OCA advises implementers of OCPP to no longer implement OCPP 2.0 and only use version 2.0.1 going forward.

Any mentions of "OCPP 2.0" refers to revision 2.0.1 unless specifically stated otherwise.

1.2. Terms and abbreviations

This section contains the terminology and abbreviations that are used throughout this document.

1.2.1. Terms

Term	Meaning
Charging Station	The Charging Station is the physical system where an EV can be charged. A Charging Station has one or more EVSEs.
Charging Station Management System (CSMS)	Charging Station Management System: manages Charging Stations and has the information for authorizing Users for using its Charging Stations.
Electric Vehicle Supply Equipment (EVSE)	An EVSE is considered as an independently operated and managed part of the Charging Station that can deliver energy to one EV at a time.
Energy Management System (EMS)	In this document this is defined as a device that manages the local loads (consumption and production) based on local and/or contractual constraints and/or contractual incentives. It has additional inputs, such as sensors and controls from e.g. PV, battery storage.

1.2.2. Abbreviations

Term	Meaning
CSO	Charging Station Operator
CSMS	Charging Station Management System
EMS	Energy Management System.
EV	Electric Vehicle
EVSE	Electric Vehicle Supply Equipment
RFID	Radio-Frequency Identification

1.3. References

Table 1. References

Reference	Description
[IEC61851-1]	IEC 61851-1 2017: EV conductive charging system - Part 1: General requirements. https://webstore.iec.ch/publication/33644
[IEC62559-2:2015]	Definition of the templates for use cases, actor list and requirements list. https://webstore.iec.ch/publication/22349
[ISO15118-1]	ISO 15118-1 specifies terms and definitions, general requirements and use cases as the basis for the other parts of ISO 15118. It provides a general overview and a common understanding of aspects influencing the charge process, payment and load leveling. https://webstore.iec.ch/publication/9272
[OCPP1.5]	http://www.openchargealliance.org/downloads/
[OCPP1.6]	http://www.openchargealliance.org/downloads/

2. New functionalities in OCPP2.0

OCPP 2.0 introduces new functionalities compared to OCPP 1.6 [[OCPP1.6](#)].

Due to improvements and new features, OCPP 2.0 is not backward compatible with OCPP 1.6 [[OCPP1.6](#)] or OCPP 1.5 [[OCPP1.5](#)].

2.1. Device Management

Device Management (also known as Device Model) is a long awaited feature especially welcomed by CSOs who manage a network of (complex) charging stations (from different vendors).

It provides the following functionality:

- Inventory reporting
- Improved error and state reporting
- Improved configuration
- Customizable Monitoring

This all should help CSOs to reduce the costs of operating a Charging Station network.

Charging Station Manufacturers are free to decide themselves how much details about a Charging Station they want to publish via Device Management: for example, they can decide what can be monitored, and what not.

2.2. Improvements for better handling of large amounts of transactions

2.2.1. One message for all transaction related functionalities

With the growing of the EV charging market, the number of Charging Stations and transactions that the CSMS needs to manage also grows. The structure and method for reporting transaction is unified in OCPP 2.0. In OCPP 1.x, the reporting of transaction data is split over the messages StartTransaction, StopTransaction, MeterValue and StatusNotification. With the market progressing towards more enhanced scheduling, a need is born for more sophisticated handling of transaction data. All the StartTransaction, StopTransaction, and transaction related MeterValue and StatusNotification messages are replaced by 'TransactionEvent'. The StatusNotification message still exists, but only for non-transaction related status notifications about connector availability.

2.2.2. Data reduction

With the introduction of JSON over Websockets in OCPP 1.6 [[OCPP1.6](#)] a great reduction of mobile data cost can be achieved. With OCPP 2.0, support for WebSocket Compression is introduced, which reduces the amount of data even more.

2.3. Improvements regarding cyber security

The following improvements have been added to harden OCPP against cyber attacks:

- Security profiles (3 levels) for Charging Station and/or CSMS authentication and Communication Security
- Key management for Client-Side certificates
- Secure firmware updates
- Security event log

2.4. Extended Smart Charging

In OCPP 2.0 Smart Charging functionality has been extended (compared to OCPP 1.6 [[OCPP1.6](#)]) to support:

- Direct Smart Charging inputs from an Energy Management System (EMS) to a Charging Station
- Improved Smart Charging with a local controller
- Support for integrated smart charging of the CSMS, Charging Station and EV ([\[ISO15118-1\]](#)).

2.5. Support for ISO 15118

The ISO 15118 standard [[ISO15118-1](#)] is a newer protocol for EVSE to EV communication, compared to IEC 61851 [[IEC61851-1](#)]. ISO 15118 allows a lot of new features and more secure communication between EVSE and EV. OCPP 2.0 supports the ISO 15118 standard, the newly added features are:

- Plug & Charge
- Smart Charging including input from the EV

2.6. Improvements for customer experience

2.6.1. More authorization options

OCPP 1.x was designed (mainly) for Charging Stations that authorize an EV driver via an RFID card/token. If other authorization systems or a mix of systems are used, the CSMS needs to know what system is used for which authorization. OCPP 2.0 has been extended to support things like: 15118 Plug & Charge [[ISO15118-1](#)], Payment Terminals, local mechanical key, Smart-phones, etc.

2.6.2. Display Messages

This provides Charging Station Operators with the possibility to configure - from the CSMS - a message on a Charging Station to be displayed to EV drivers. Messages can be transaction related or global.

2.6.3. EV Driver preferred languages

To be able to show messages to an EV driver in a language the driver understands best, OCPP 2.0 provides the possibility to send the language preference of a driver to a Charging Station.

2.6.4. Tariff and Costs

OCPP 2.0 allows Charging Stations to show the applicable tariff/price before an EV driver starts charging, to show the running total cost during a charging transaction and/or to show the final total cost after the transaction is finished.

2.7. Transport Protocols: OCPP-J Improvements

2.7.1. Simple Message routing

A description has been added on how to create a simple solution for OCPP message routing in, for example, a Local Controller. This is defined in Part 4, Section 6: OCPP Routing.

2.7.2. No SOAP Support

OCPP 2.0 no longer supports SOAP as a transport protocol. This decision was taken by the OCA members, who believe that the protocol does no longer lend itself for constrained computing resources that many Charging Stations operate under. The verbosity of the protocol could lead to slower performance and requires a higher bandwidth, which, in many cases, leads to higher cellular costs. SOAP is also difficult to support when communication is via local site networking.

2.8. Minor changes/extensions

2.8.1. Renamed messages

In the OCPP 1.x series, the names of all messages were kept unchanged for backward compatibility, even though some message names were found to be confusing or misleading in practice. In OCPP 2.0 message names have been changed, where appropriate, to improve clarity and understanding.

Example: RemoteStartTransaction.req: a lot of implementers thought it meant the Charging Station should start the transaction, but in fact it is a request to try to start a transaction. However, for example, if no cable is plugged in, no transaction can be started. Since the message was always intended to be a request, it has been changed to a more logical name: RequestStartTransactionRequest.

2.8.2. TransactionId Identification & Message Sequencing

In OCPP 2.0, transaction identifiers are generated by the Charging Station, to facilitate offline charging sessions, in contrast to OCPP 1.x, where transaction identifiers were generated at the CSMS and sent to the Charging Station. In addition, all messages relating to a transaction are assigned incremental sequence numbering, to facilitate transaction data completeness checking at the CSMS.

2.8.3. Extended enumerations

Many enumerations have been extended to support more use cases, provide more options etc.

2.8.4. Offline Transaction Event Indication

Charging Stations can optionally indicate in transaction messages that a transaction event occurred while the Charging Station was Offline. This can assist a CSMS with the processing of transactions.

2.8.5. Personal message

Message that can be shown to the EV Driver and can be used for tariff information, user greetings and for indicating why a driver is not authorized to charge. When a driver uses an authorization method (RFID for example) and the CSMS does not authorize the driver to start charging, this field can thus contain additional reasons to provide the driver with a meaningful explanation why (s)he is not allowed to charge.

3. OCPP 2.0 Documentation Structure

3.1. Overview of Specification Parts

The overall structure of the standard has been improved, making the new specification easier to read, implement and test.

For readability and implementation purposes, OCPP 2.0 is divided in seven parts.

Table 2. Parts

Part 0	Introduction (this document)
Part 1	Architecture & Topology
Part 2	Specification: Use Cases and Requirements, Messages, Data Types and Referenced Components and Variables Appendices: Security Events, Standardized Units of Measure, Components and Variables
Part 3	Schemas
Part 4	Implementation Guide JSON
Part 5	Certification Profiles
Part 6	Test Cases

In contrast to OCPP 1.6 [OCPP1.6], the OCPP 2.0 specification is written in a different structure, based on [IEC62559-2:2015]: "Use case methodology - Part 2: Definition of the template for use cases, actor list and requirements list".

Part 2, the specification, is divided into 'Functional Blocks'. These Functional Blocks contain use cases and requirements. Messages, Data Types and Referenced Components and Variables are described at the end of the document. The Appendices can be found in the separate document: Part 2 - Appendices.

Messages and Data Types are structured in almost the same way as the previous OCPP specification [OCPP1.6].

3.2. Functional Blocks

OCPP 2.0 consists of the following Functional Blocks.

Table 3. Functional Blocks

Clause	Functional Block Title	Description
A.	Security	This Functional Block describes a security specification for the OCPP protocol.
B.	Provisioning	This Functional Block describes all the functionalities that help a CSO provision their Charging Stations, allowing them to be registered and accepted on their network and retrieving basic configuration information from these Charging Stations.
C.	Authorization	This Functional Block describes all the authorization related functionality: AuthorizeRequest message handling/behavior and Authorization Cache functionality.
D.	Local Authorization List Management	This Functional Block describes functionality for managing the Local Authorization List.
E.	Transactions	This Functional Block describes the basic OCPP Transaction related functionality for transactions that are started/stopped on the Charging Station.
F.	Remote Control	This Functional Block describes three types of use cases for remote control management from the CSMS: Remote Transaction Control, Unlocking a Connector and Remote Trigger.
G.	Availability	This functional Block describes the functionality of sending status notification messages.
H.	Reservation	This Functional Block describes the reservation functionality of a Charging Station.
I.	Tariff and Cost	This Functional Block provides tariff and cost information to an EV Driver, when a Charging Station is capable of showing this on a display. Before a driver starts charging tariff information needs to be given, detailed prices for all the components that make up the tariff plan applicable to this driver at this Charging Station. During charging the EV Driver needs to be shown the running total cost, updated at a regular, fitting interval. When the EV Driver stops charging the total cost of this transaction needs to be shown.
J.	Metering	This Functional Block describes the functionality for sending meter values, on a periodic sampling and/or clock-aligned timing basis.
K.	Smart Charging	This Functional Block describes all the functionality that enables the CSO (or indirectly a third party) to influence the charging current/power of a charging session, or set limits to the amount of power/current a Charging Station can offer to an EV.
L.	Firmware Management	This Functional Block describes the functionality that enables a CSO to update the firmware of a Charging Station.
M.	ISO 15118 Certificate Management	This Functional Block provides the installation and update of ISO 15118 certificates.
N.	Diagnostics	This Functional Block describes the functionality that enables a CSO to request and track the upload of a diagnostics file from a Charging Station, and to manage the monitoring of Charging Station data.
O.	Display Message	With the DisplayMessage feature OCPP enables a CSO to display a message on a Charging Station, that is not part of the firmware of the Charging Station. The CSO gets control over these messages: the CSO can set, retrieve (get), replace and clear messages.
P.	Data Transfer	This Functional Block describes the functionality that enables a party to add custom commands to OCPP, enabling custom extension to OCPP.

3.3. All Functional Blocks and use cases

The following table shows the full list of use cases supported by OCPP 2.0 and which use cases were already supported by OCPP 1.6 [\[OCPP1.6\]](#).

Clause	Functional Block	UC ID	Use case name	OCPP 1.6	New in OCPP 2.0
A	Security	A01	Update Charging Station Password for HTTP Basic Authentication		o
		A02	Update Charging Station Certificate by request of CSMS		o
		A03	Update Charging Station Certificate initiated by the Charging Station		o
		A04	Security Event Notification		o
B	Provisioning	B01	Cold Boot Charging Station	o	
		B02	Cold Boot Charging Station - Pending	o	
		B03	Cold Boot Charging Station - Rejected	o	
		B04	Offline Behavior Idle Charging Station	o	
		B05	Set Variables		o
		B06	Get Variables		o
		B07	Get Base Report		o
		B08	Get Custom Report		o
		B09	Setting a new NetworkConnectionProfile		o
		B10	Migrate to new CSMS		o
		B11	Reset - Without Ongoing Transaction	o	
		B12	Reset - With Ongoing Transaction	o	
C	Authorization	C01	EV Driver Authorization using RFID	o	
		C02	Authorization using a start button		o
		C03	Authorization using credit/debit card		o
		C04	Authorization using PIN-code		o
		C05	Authorization for CSMS initiated transactions		o
		C06	Authorization using local id type		o
		C07	Authorization using Contract Certificates		o
		C08	Authorization at EVSE using ISO 15118 External Identification Means (EIM)		o
		C09	Authorization by GroupId	o	
		C10	Store Authorization Data in the Authorization Cache	o	
		C11	Clear Authorization Data in Authorization Cache	o	
		C12	Start Transaction - Cached Id	o	
		C13	Offline Authorization through Local Authorization List	o	
		C14	Online Authorization through Local Authorization List	o	
		C15	Offline Authorization of unknown Id	o	
		C16	Stop Transaction with a Master Pass		o
D	LocalAuthorizationList	D01	Send Local Authorization List	o	
		D02	Get Local List Version	o	
E	Transactions	E01	Start Transaction Options		o
		E02	Start Transaction - Cable Plugin First	o	
		E03	Start Transaction - IdToken First	o	
		E04	Transaction started while Charging Station is offline	o	

Clause	Functional Block	UC ID	Use case name	OCPP 1.6	New in OCPP 2.0
		E05	Start Transaction - Id not Accepted	o	
		E06	Stop Transaction Options		o
		E07	Transaction locally stopped by IdToken	o	
		E08	Transaction stopped while Charging Station is offline	o	
		E09	When cable disconnected on EV-side: Stop Transaction	o	
		E10	When cable disconnected on EV-side: Suspend Transaction	o	
		E11	Connection Loss During Transaction	o	
		E12	Inform CSMS of an Offline Occurred Transaction	o	
		E13	Transaction related message not accepted by CSMS	o	
		E14	Check transaction status		o
		E15	End of charging process	o	
F	RemoteControl	F01	Remote Start Transaction - Cable Plugin First	o	
		F02	Remote Start Transaction - Remote Start First	o	
		F03	Remote Stop Transaction	o	
		F04	Remote Stop ISO 15118 charging from CSMS		o
		F05	Remotely Unlock Connector	o	
		F06	Trigger Message	o	
G	Availability	G01	Status Notification	o	
		G02	Heartbeat	o	
		G03	Change Availability EVSE	o	
		G04	Change Availability Charging Station	o	
		G05	Lock Failure	o	
H	Reservation	H01	Reservation	o	
		H02	Cancel Reservation	o	
		H03	Use a reserved EVSE	o	
		H04	Reservation Ended, not used	o	
I	Tariff and Costs	I01	Show EV Driver-specific tariff information		o
		I02	Show EV Driver running total cost during charging		o
		I03	Show EV Driver final total cost after charging		o
		I04	Show fallback tariff information		o
		I05	Show fallback total cost message		o
		I06	Update Tariff Information During Transaction		o
J	Metering	J01	Sending Meter Values not related to a transaction	o	
		J02	Sending transaction related Meter Values	o	
		J03	Charging Loop with metering information exchange		o
K	SmartCharging	K01	SetChargingProfile	o	
		K02	Central Smart Charging	o	
		K03	Local Smart Charging	o	
		K04	Internal Load Balancing	o	
		K05	Remote Start Transaction with Charging Profile	o	
		K06	Offline Behavior Smart Charging During Transaction	o	
		K07	Offline Behavior Smart Charging at Start of Transaction	o	
		K08	Get Composite Schedule	o	

Clause	Functional Block	UC ID	Use case name	OCPP 1.6	New in OCPP 2.0
		K09	Get Charging Profiles		o
		K10	Clear Charging Profile	o	
		K11	Set / Update External Charging Limit With Ongoing Transaction		o
		K12	Set / Update External Charging Limit Without Ongoing Transaction		o
		K13	Reset / release external charging limit		o
		K14	External Charging Limit with Local Controller		o
		K15	Charging with load leveling based on High Level Communication		o
		K16	Optimized charging with scheduling to the CSMS		o
		K17	Renegotiating a Charging Schedule		o
L	Firmware Management	L01	Secure Firmware Update		o
		L02	Non-Secure Firmware Update	o	
		L03	Publish Firmware file on Local Controller		o
		L04	Unpublish Firmware file on Local Controller		o
M	ISO 15118 Certificate Management	M01	Certificate Installation EV		o
		M02	Certificate Update EV		o
		M03	Retrieve list of available certificates from a Charging Station		o
		M04	Delete a specific certificate from a Charging Station		o
		M05	Install CA certificate in a Charging Station		o
		M06	Get Charging Station Certificate status		o
N	Diagnostics	N01	Retrieve Log Information	o	
		N02	Get Monitoring report		o
		N03	Set Monitoring Base		o
		N04	Set Variable Monitoring		o
		N05	Set Monitoring Level		o
		N06	Clear / Remove Monitoring		o
		N07	Alert Event		o
		N08	Periodic Event		o
		N09	Get Customer Information		o
		N10	Clear Customer Information		o
O	Display Message	O01	Set DisplayMessage		o
		O02	Set DisplayMessage for Transaction		o
		O03	Get All DisplayMessages		o
		O04	Get Specific DisplayMessages		o
		O05	Clear a DisplayMessage		o
		O06	Replace DisplayMessage		o
P	DataTransfer	P01	Data Transfer to the Charging Station	o	
		P02	Data Transfer to the CSMS	o	

NOTE

OCPP is used in many different regions and for many different charging solutions. Not all functionalities offered by OCPP 2.0 will be applicable to all implementations. Implementers can decide what specific functionalities apply to their charging solution.

For interoperability purposes, the Open Charge Alliance introduces Certification Profiles in Part 5 of the specification.

4. Basic implementation of OCPP 2.0

This section is informative.

The OCPP protocol describes a large number of use cases and messages, which are not all needed to implement a basic Charging Station or CSMS. The table below lists messages that are typically implemented to deliver basic functionality for an OCPP managed Charging Station. The purpose of this list is to guide developers that are new to OCPP.

Please note: this table does *not* define what needs to be done to become OCPP 2.0 "certified". The functionality that is to be implemented to become OCPP 2.0 certified is described in Part 5 of the specification, "Certification Profiles".

Table 4. OCPP 2.0 Basic Implementation

Functionality	Use cases	Messages
Booting a Charging Station	B01-B04	BootNotification
Configuring a Charging Station	B05-B07	SetVariables, GetVariables and GetReportBase (respond correctly to requests with reportBase = ConfigurationInventory, FullInventory and SummaryInventory).
Resetting a Charging Station	B11-B12	Reset
Authorization options	One of C01, C02 and C04	Authorize
Transaction mechanism	E01 (one of S1-S6), E02-E03, E05, E06 (one of S1-S6), E07-E08, One of E09-E10, E11-E13	TransactionEvent
Availability	G01, G03-G04	Only ChangeAvailability and StatusNotification.
Monitoring Events	G05, N07	A basic implementation of the NotifyEvent message to be used to report operational state changes and problem/error conditions of the Charging Station, e.g. for Lock Failure. Also used for reporting built-in monitoring events.
Sending transaction related Meter values	J02	TransactionEvent
DataTransfer	P01-P02	Any OCPP implementations should at least be able to reject any request for DataTransfer if no (special) functionality is implemented.

NOTE

Please also refer to the section on Minimum Device Model in part 1.



OCPP 2.0.1

Part 1 - Architecture & Topology

FINAL, 2020-03-31

Table of Contents

Disclaimer	1
Version History	2
1. Introduction.....	3
1.1. Goal of this document	3
1.2. Terms and abbreviations.....	3
2. 3-tier model.....	4
3. Information Model	5
4. Device Model: Addressing Components and Variables.....	6
4.1. Components	6
4.2. Variables	7
4.3. Characteristics and Attributes	7
4.4. Monitoring	9
4.5. Standardized lists of Components and Variables	10
4.6. Minimum Device Model.....	10
5. Information Model vs. Device Model.....	11
6. Using OCPP for other purposes than EV charging.....	12
7. Numbering	13
7.1. EVSE numbering.....	13
7.2. Connector numbering	13
7.3. Transaction IDs.....	13
8. Topologies supported by OCPP	14
8.1. Charging Station(s) directly connected to CSMS.....	14
8.2. Multiple Charging Stations connected to CSMS via Local Proxy	14
8.3. Multiple Charging Stations connected to CSMS via Local Controller	15
8.4. Non-OCPP Charging Stations connected to CSMS via OCPP Local Controller.....	15
8.5. DSO control signals to CSMS	15
8.6. Parallel control by CSMS and EMS.....	16
9. Part 1 Appendix: OCPP Information Model.....	17
9.1. Explanation of UML representation and message generation	17
9.2. Visual Representation of OCPP Information Model.....	18

Disclaimer

Copyright © 2010 – 2020 Open Charge Alliance. All rights reserved.

This document is made available under the **Creative Commons Attribution-NoDerivatives 4.0 International Public License** (<https://creativecommons.org/licenses/by-nd/4.0/legalcode>).

Version History

Version	Date	Author	Description
2.0.1	2020-03-31	Franc Buve (OCA) Milan Jansen (OCA) Paul Klapwijk (OCA)	Final version of OCPP 2.0.1
2.0	2018-04-11	Milan Jansen (OCA) Paul Klapwijk (OCA) Robert de Leeuw (<i>IHomer</i>) Robben Riksen (<i>Alliander</i>)	OCPP 2.0 April 2018 First release of this Architecture & Topology document

1. Introduction

1.1. Goal of this document

The goal of this document is to describe a number of architecture related topics for OCPP 2.0.1.

OCPP was originally intended for two way communication between a backoffice, in OCPP the *Charging Station Management System* (in this document: CSMS) and a Charging Station. The protocol has become more advanced and with every new revision new functionalities and options are added. It has evolved into a protocol that can be used in different architectures for different types of Charging Stations.

This document describes, in addition to the original "simple" setup CSMS <> Charging Station, a number of topologies as an additional explanation for using OCPP. Furthermore, the Device Management concept to configure and monitor any type of Charging Station, the OCPP Information Model and the 3-tier model are explained.

This document is partially **informative** and partially **normative** and is not intended to limit the use of OCPP. However, it does add an explanation what kind of use of OCPP the creators of OCPP had in mind when creating this version of the specification. This document is therefore also intended to support the reader of the protocol specification in Part 2 of OCPP to understand how it can be used.

1.2. Terms and abbreviations

This section contains the terminology and abbreviations that are used throughout this document.

1.2.1. Terms

Term	Meaning
Charging Station	The Charging Station is the physical system where EVs can be charged. A Charging Station has one or more EVSEs.
Connector	The term Connector, as used in this specification, refers to an independently operated and managed electrical outlet on a Charging Station. In other words, this corresponds to a single physical Connector. In some cases an EVSE may have multiple physical socket types and/or tethered cable/Connector arrangements(i.e. Connectors) to facilitate different vehicle types (e.g. four-wheeled EVs and electric scooters).
EVSE	An EVSE is considered as an independently operated and managed part of the Charging Station that can deliver energy to one EV at a time.
Local port Smart Meter	The Local port on a Smart Meter is a port (for example serial) on a digital electricity meter that provides access to information about meter readings and usage.

1.2.2. Abbreviations

Abbreviation	Meaning
DSO	Distribution System Operator
CSO	Charging Station Operator
CSMS	Charging Station Management System
EMS	Energy Management System. In this document this is defined as a device that manages the local loads (consumption an production) based on local and/or contractual constraints and/or contractual incentives. It has additional inputs, such as sensors and controls from e.g. PV, battery storage.
EVSE	Electric Vehicle Supply Equipment
LC	Local Controller. In this document this is defined as a device that can send messages to its Charging Stations, independently of the CSMS. A typical usage for this is the local smart charging case described in the Smart Charging chapter of Part 2 of OCPP, where a Local Controller can impose charge limits on its Charging Stations.
LP	Local Proxy. Acts as a message router.

2. 3-tier model

This section is informative.

To understand the terminology in the OCPP specification, it is important to understand the starting point of this specification. The OCPP specification uses the term Charging Station as the physical system where EVs can be charged. A Charging Station can have one or more EVSEs (Electric Vehicle Supply Equipment). An EVSE is considered as a part of the Charging Station that can deliver energy to one EV at a time. The term Connector, as used in this specification, refers to an independently operated and managed electrical outlet on a Charging Station, in other words, this corresponds to a single physical Connector. In some cases an EVSE may have multiple physical socket types and/or tethered cable/connector arrangements to facilitate different vehicle types (e.g. four-wheeled EVs and electric scooters). This setup is referred to as the 3-tier model and visualized in the figure below.

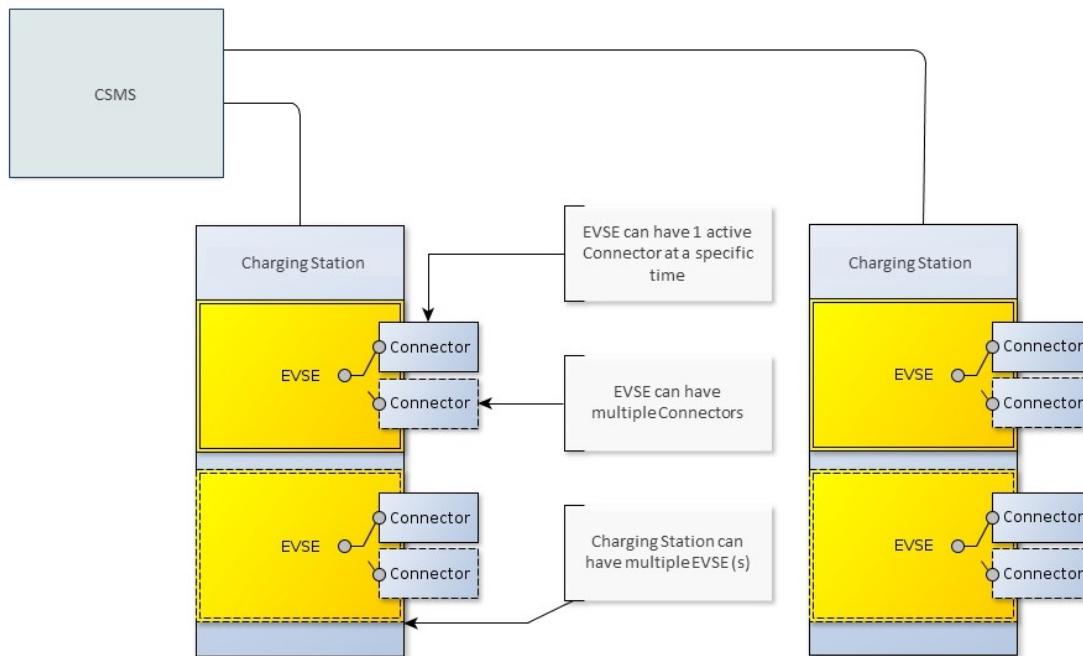


Figure 1. 3-tier model as used in OCPP

NOTE

This section describes the charging infrastructure on a logical level for communication purposes. We do not wish to impose a mapping onto physical hardware. This is a manufacturer's choice. For example, the EVSE might be integrated into a Charging Station and to look as just a part of that device, but it might just as well have its own casing and live outside of the physical entity Charging Station, for example a charging plaza with 20 EVSEs and Connectors which communicates via 1 modem as 1 Charging Station to the CSMS is seen by OCPP as 1 Charging Station.

3. Information Model

This section is informative.

Given the growing complexity of the messages of OCPP, OCPP 2.0.1 is based on an *Information Model* as a blueprint for the messages and inherent schemas of OCPP. With an information model, we mean a logical object set, describing real objects with all their properties. This provides an informative representation of information structure in the protocol. Furthermore, it enables making objects within OCPP reusable and enables consistent definition of messages and automatically generated message schemas (Part 3).

The Information Model is a model, also called Domain Model or Core Model, based on which the OCPP messages and datatypes are generated. These datatypes are extracted from the the OCPP 1.6 specification and are named Core DataTypes and Qualified DataTypes. The figure below illustrates how the DataTypes in the information model are built up.

In part 2 - Specification, chapter Datatypes, some DataTypes have the Common: prefix. This originates from the Information Model. It means that the DataType is able to be shared among other DataTypes and Messages. This has no impact on the OCPP implementation of a device.

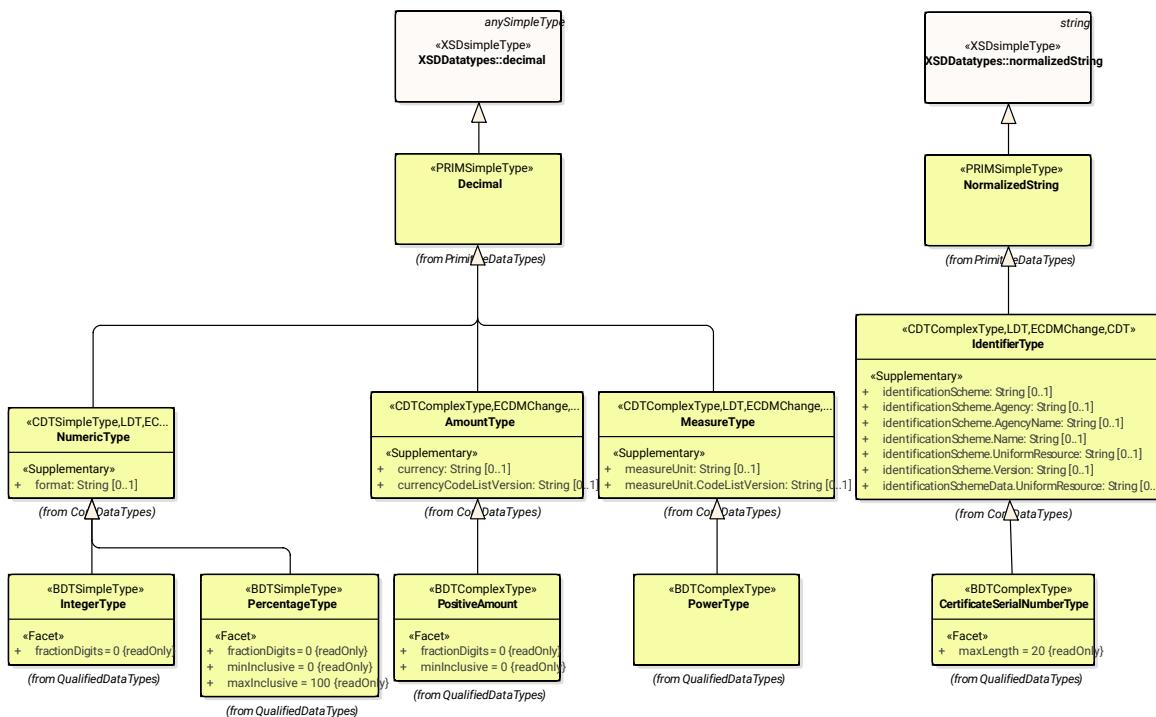


Figure 2. Example datatypes

The Information Model is divided into a number of "functions" to have a better overview of the model (thus for readability):

- Transactions
- SmartCharging
- Metering
- Security (Profiles/Authorization)
- Communication
- SecondaryActorSchedule

For more details about the actual model per function, please refer to the appendix.

4. Device Model: Addressing Components and Variables

The Device Model refers to a generalized mechanism within OCPP to enable any model of Charging Station to report how it is build up, so it can be managed from any CSMS. To manage a Charging Station with the Device Model (i.e. "to manage a device") a number of messages and use cases is defined to configure and monitor a Charging Station in detail, without defining the structure of the Charging Station in advance. To be able do do this, OCPP provides a generalized mechanism to allow the exchange of a wide range of information about Charging Station. This version of the Device Model has the 3-tier model (Charging Station, EVSE, Connector) as its starting point, which means that any description created with the Device Model follows these three tiers. The remainder of this chapter describes how the data (and associated meta-data) looks like that can be exchanged between a Charging Station and a CSMS. The use cases and messages that are used to manage a device are *not* described here, but in Part 2 of the specification. This chapter only focuses on the data model.

4.1. Components

In OCPP 2.0.1, a Charging Station is modelled as a set of "*Components*", typically representing physical devices (including any external equipment to which it is connected for data gathering and/or control), logical functionality, or logical data entities. *Components* of different types are primarily identified by a *ComponentName*, that is either the name of a *standardized component* (see OCPP part 2c), or a custom/non-standardized component name, for new, pre-standardized equipment, vendor specific extensions, etc.

ChargingStation (TopLevel), *EVSE*, and *Connector* represent the three major "tiers" of a Charging Station, and constitute an implicit "location-based" addressing scheme that is widely used in many OCPP data structures.

By default, all *components* are located at the *ChargingStation* tier, but individual instances of any component can be associated with a specific *EVSE*, or a specific *Connector* (on a specific *EVSE*) by including *EVSE* or *EVSE* and *Connector* identification numbers as part of a component addressing reference.

Additionally, there can be more than one instance of a component (in the functional dimension), representing multi-occurrence physical or logical components (e.g. power converter modules, fan banks, resident firmware images, etc.).

Each distinct *component* instance is uniquely identified by an (optional) *componentInstance* addressing key. When no *componentInstance* is provided, then the default or only instance of a *component* is referenced.

Components do not in themselves hold data: all externally accessible data associated with each component instance is represented by a set of *variables* that can be read, set, and/or monitored for changes. The relationship of a Component with one or more Variables is illustrated in below.

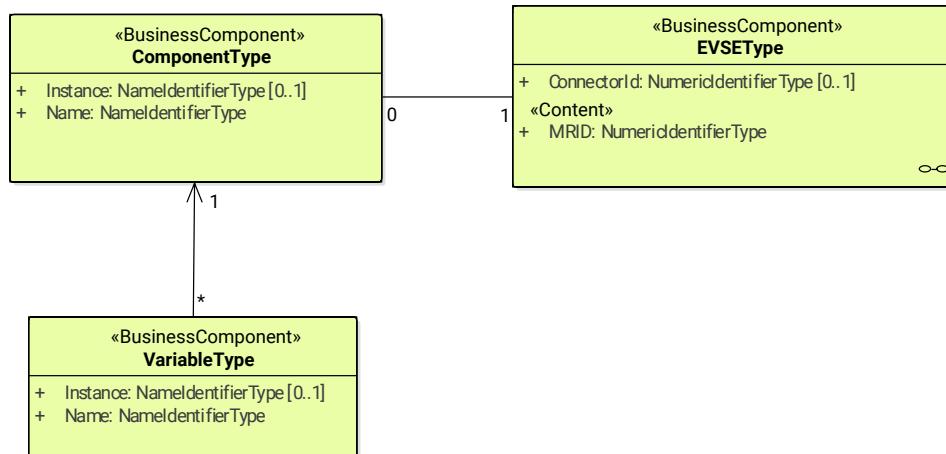


Figure 3. Component and variables

The table below illustrates some common components (by their standardized component-names), and examples of the hierarchical location levels at which they typically occur for a basic home charger and a typical public Charging Station.

Basic home charger example configuration		
ChargingStation tier	EVSE tier	Connector tier
ChargingStation (itself, as a whole)	EVSE (itself, as a whole)	Connector (itself, as a whole)
RadioLink	ControlMetering	PlugRetentionLock
TokenReader	OverCurrentBreaker	
Controller	RCD	
	ChargingStatusIndicator	

Public Charging Station example configuration		
ChargingStation tier	EVSE tier	Connector tier
ChargingStation (itself, as a whole)	EVSE (itself, as a whole)	Connector (itself, as a whole)
ElectricalFeed	ElectricalFeed	AccessProtection
TokenReader	TokenReader	PlugRetentionLock
Display	Display	
FiscalMetering	FiscalMetering	
Clock	ControlMetering	
Controller	OverCurrentBreaker	
	RCD	
	ChargingStatusIndicator	

4.2. Variables

Every *component* has a number of *variables*, that can, as appropriate, be used to hold, set, read, and/or report on all (externally visible) data applicable to that *component*, including configuration parameters, measured values (e.g. a current or a temperature) and/or monitored changes to variable values.

Although many *components* can have associated *variables* that are, by their nature, specific to the component type (e.g. *ConnectorType* for a *Connector* component), there are a minimal set of standardized *variables* that are used to provide standardized high level event notification and state/status reporting (e.g. *Problem*, *Active*) on a global and/or selective basis, and also to report component presence, availability, etc. during the inventorying/discovery process (e.g. *Available*, *Enabled*). A Charging Station is not required to report the base variables: *Present*, *Available* and *Enabled* when they are readonly and set to *true*. When a Charging Station does not report: *Present*, *Available* and/or *Enabled* the Central System SHALL assume them to be readonly and set to *true*. Variables can be any of a range of common general-purpose data types (boolean, integer, decimal, date-time, string), but also can have their allowable values constrained to particular ranges, enumeration lists, sets, or ordered lists.

To support complex components, there can be more than one instance of any given variable name associated with any components (e.g. power converter modules reporting temperature, current, or voltage at multiple points).

Each distinct *variable* instance is uniquely identified by an (optional) *variableInstance* addressing key string value. When no *variableInstance* is provided, then the default or only instance of a *variable* is referenced.

4.3. Characteristics and Attributes

Each *variable*, in addition to its primary ("Actual") value, can have a set of associated secondary data that is linked to the same primary *variable* name and *variableInstance*.

This greatly avoids cluttering the *variables* namespace with confusing clusters of ancillary variable names (e.g. *FanSpeed*, *FanSpeedUnits*, *MinimumFanSpeed*, *BaseFanSpeed*) that lack consistence and discoverability.

The ancillary variable data includes:

- Variable characteristics meta-data (read-only)
 - Unit of measure (V,W,kW,kWh, etc.)
 - Data type (Integer, Decimal, String, Date, OptionList, etc.)
 - Lower limit
 - Upper limit
 - List of allowed values for enumerated variables
- Variable attributes (read-write):
 - Actual value
 - Target value
 - Configured lower limit
 - Configured upper limit
 - Mutability (whether the value can be altered or not, e.g. *ReadOnly* or *ReadWrite*)
 - Persistence (whether the value is preserved in case of a reboot or power loss)

The relationship of a Variable with one or more VariableAttributes is illustrated in the figure below.

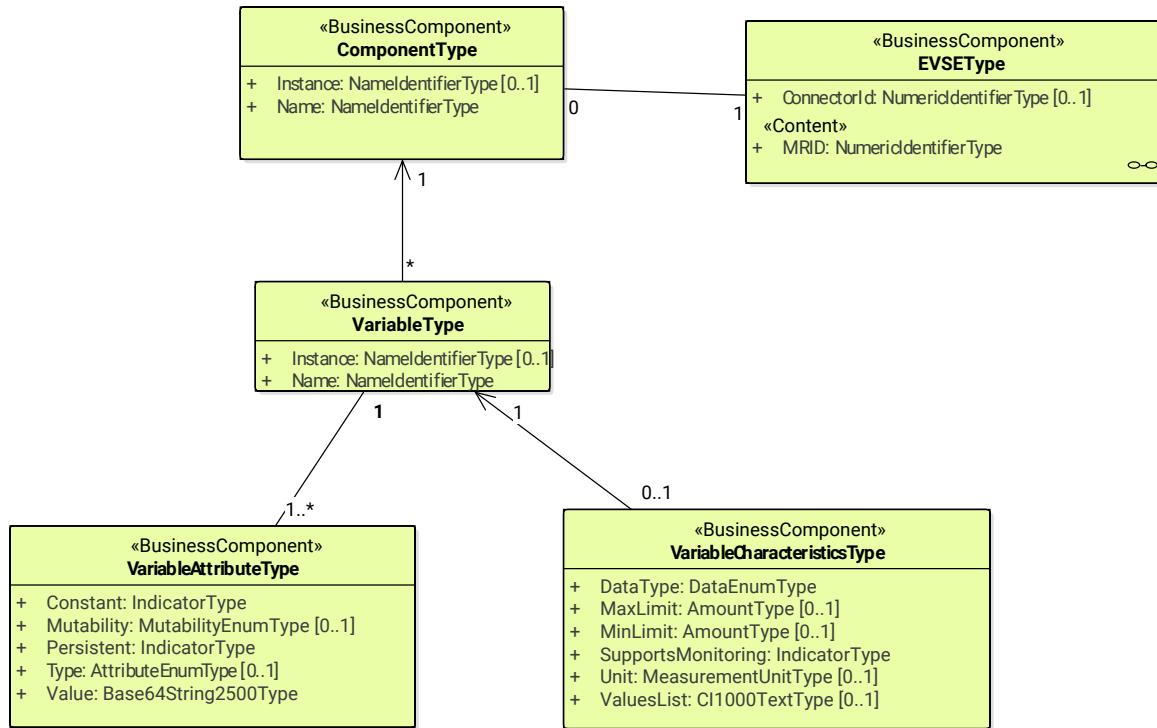


Figure 4. Variable attributes and characteristics

There is a difference between how to implement (physical) devices and (virtual) controller components, using the DeviceModel. A (virtual) controller component has to be implemented as described in part 2 chapter the "Referenced Components and Variables". These kind of components/variables are only using the variableAttribute type 'Actual'. Depending on if this variableAttribute is writable, the CSMS can use this to set a new value.

(Physical) devices are a bit more complex to implement. For example, there is a fan with a fan speed, that has a (physical) limit with a range of 0 - 1000. But it should not be allowed to set the value below 200, because the fan can stop functioning. And it should not be set above 500, because that would be bad for the fan on the long run. When implementing this device using the DeviceModel, it can be defined as follows:

Component	name	Fan					
Variable	name	FanSpeed					
	variableAttribute 1	<table border="1"> <tr> <td>type</td><td>Actual</td></tr> <tr> <td>value</td><td><The current fan speed value of the fan.></td></tr> <tr> <td>mutability</td><td>ReadOnly</td></tr> </table>	type	Actual	value	<The current fan speed value of the fan.>	mutability
type	Actual						
value	<The current fan speed value of the fan.>						
mutability	ReadOnly						
variableAttribute 2	<table border="1"> <tr> <td>type</td><td>Target</td></tr> <tr> <td>value</td><td><The CSMS can use this value to adjust the fan speed. The Charging Station SHALL try to keep the actual value at the target value.></td></tr> <tr> <td>mutability</td><td>ReadWrite</td></tr> </table>	type	Target	value	<The CSMS can use this value to adjust the fan speed. The Charging Station SHALL try to keep the actual value at the target value.>	mutability	ReadWrite
type	Target						
value	<The CSMS can use this value to adjust the fan speed. The Charging Station SHALL try to keep the actual value at the target value.>						
mutability	ReadWrite						
variableAttribute 3	<table border="1"> <tr> <td>type</td><td>MaxSet</td></tr> <tr> <td>value</td><td><The value '500' from the example. The target may not be set above this value.></td></tr> </table>	type	MaxSet	value	<The value '500' from the example. The target may not be set above this value.>		
type	MaxSet						
value	<The value '500' from the example. The target may not be set above this value.>						
variableAttribute 4	<table border="1"> <tr> <td>type</td><td>MinSet</td></tr> <tr> <td>value</td><td><The value '200' from the example. The target may not be set below this value.></td></tr> </table>	type	MinSet	value	<The value '200' from the example. The target may not be set below this value.>		
type	MinSet						
value	<The value '200' from the example. The target may not be set below this value.>						
variableCharacteristics	<table border="1"> <tr> <td>maxLimit</td><td><The value '1000' from the example. This could be the physical max limit of the fan.></td></tr> <tr> <td>minLimit</td><td><The value '0' from the example. This could be the physical min limit of the fan. This could also be -1000, if the fan is also able to rotate in the other direction.></td></tr> </table>	maxLimit	<The value '1000' from the example. This could be the physical max limit of the fan.>	minLimit	<The value '0' from the example. This could be the physical min limit of the fan. This could also be -1000, if the fan is also able to rotate in the other direction.>		
maxLimit	<The value '1000' from the example. This could be the physical max limit of the fan.>						
minLimit	<The value '0' from the example. This could be the physical min limit of the fan. This could also be -1000, if the fan is also able to rotate in the other direction.>						
Description	This is an example of how a fan could be defined using the DeviceModel.						

When trying to set the target with value 600, the Charging Station will first check the allowed min and max values/limits and reject the set. If the target value is set to 500, the value is within range and the Charging Station will allow the set and start to adjust the actual fan speed. If the actual fan speed is measured to be 502, it's out of range. But it should be reported to the CSMS, so the

actual value of a physical component should be updated without checking the min and max values/limits.

4.4. Monitoring

Optional monitoring settings can be associated with a variable, that allow changes to *variable (Actual)* values are to be reported to the CSMS as event notifications.

These include:

- Monitoring value
- Monitoring type: upper threshold, lower threshold, delta, periodic
- Severity level when reporting the event

The following table show which MonitorType/dataType combinations are possible.

	string	decimal	integer	dateTime	boolean	OptionList	SequenceList	MemberList
UpperThreshold		X	X					
LowerThreshold		X	X					
Delta	X	X	X	X	X	X	X	X
Periodic	X	X	X		X	X	X	X
PeriodicClockAligned	X	X	X		X	X	X	X

- For *UpperThreshold* and *LowerThreshold* the value represents the to be exceeded value by the actual value of the variable.
- For *Delta* this value represents the change in value comparing with the actual value from the moment the monitor was set.
 - When the dataType of the variable is integer or decimal, this value represents the difference to be reached to trigger the monitor.
 - When the dataType of the variable is dateTime the unit of measure will be in seconds.
 - When the dataType of the variable is string, boolean, OptionList, SequenceList or MemberList, this value is ignored. The monitor will be triggered by every change in the actual value.
- When a delta monitor is triggered OR when the Charging Station has rebooted, the Charging Station shall set a new momentary value.
- For *Periodic* and *PeriodicClockAligned* the value represents the interval in seconds.

The relationship between a Variable and one or more VariableMonitoring elements is illustrated in the figure below.

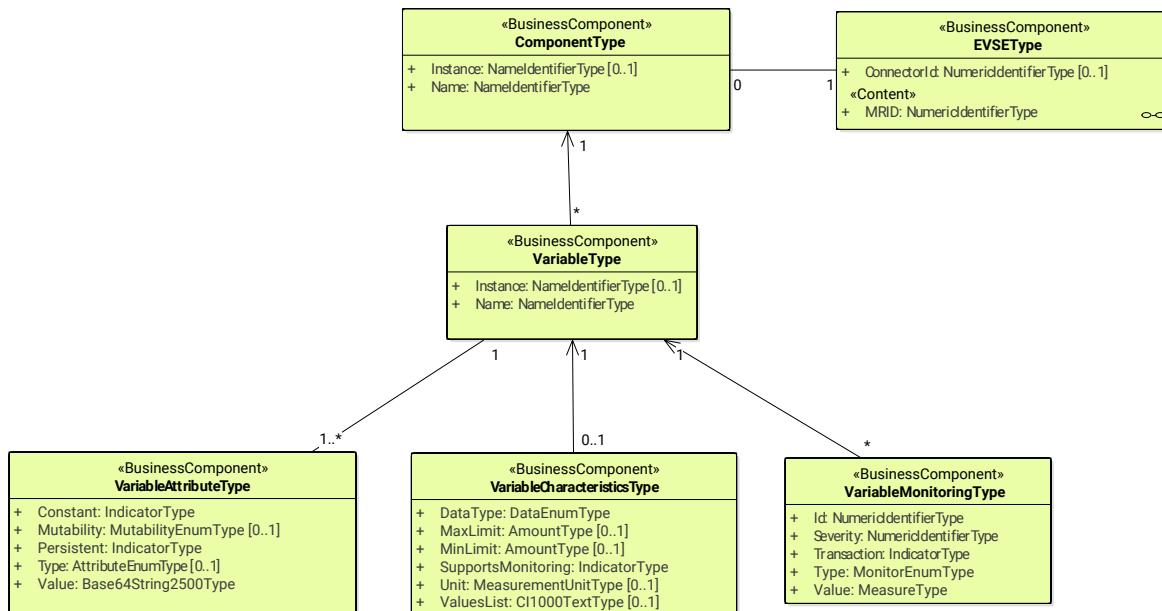


Figure 5. Variables and monitoring

4.5. Standardized lists of Components and Variables

To provide some level of interoperability between different Charging Stations and CSMSs, besides the above defined model of *Components* and *Variables*, part 2 - appendices of the OCPP specification provides a list of standardized names for Components and Variables. The idea of this lists is to make sure that *if* a Charging Station and CSMS want to exchange information about a component, they both use the same name and description *if* it is listed in the OCPP specification. For names of a *Components* or *Variables* that are not listed in the specification, bilateral appointments between Charging Station manufacturer and CSMS are to be made. In these cases it is advised to provide feedback to the Open Charge Alliance to be able to include new/additional *Components* and *Variables* in new versions of OCPP.

4.6. Minimum Device Model

Since the Device Model is a *generalized* mechanism which can be applied to any model of Charging Station, the complexity of different implementations can vary. It consists of a number of use cases and messages that are not all required. This section describes the minimum part of the Device Model that needs to be implemented to create a working implementation of OCPP 2.0.1.

The Device Model introduces Components and Variables that can be used for configuring and monitoring a Charging Station. A number of these Components and Variables are included in the list of *Referenced Components and Variables* (grouped by Functional Block) in Part 2 of the specification. When implementing a Functional Block, ALL required Configuration Variables that belong to a Functional Block SHALL be implemented. The required Configuration Variables from the General section SHALL also be implemented for all implementations of OCPP 2.0.1.

The following table describes which messages are required or optional to implement for all use cases that are part of the Device Model implementation.

Use cases / messages that are part of a minimum Device Model implementation	
Use case	Messages
B05 Set Variables	SetVariables message MUST be implemented
B06 Get Variables	GetVariables message MUST be implemented.
B07 Get Base Report	GetBaseReport message MUST be implemented and MUST support all 3 'ReportBases'. The content of these reports depends on the implementation of the Charging Station. It is up to the implementer to decide which components and variables exist in the implementation.
Additional use cases / messages that are <i>not</i> part of a minimum Device Model implementation	
Use case	Messages
B08 Get Custom Report	GetCustomReport message is optional.
N02 Get Monitoring Report	GetMonitoringReportRequest message is optional.
N03 Set Monitoring Base	SetMonitoringBaseRequest message is optional.
N04 Set Variable Monitoring	SetVariableMonitoringRequest message is optional.
N05 Set Monitoring Level	SetMonitoringLevelRequest message is optional.
N06 Clear/Remove Monitoring	ClearVariableMonitoringRequest message is optional.
N07 Alert Event	it is RECOMMENDED that NotifyEventRequest is implemented in the Charging Station even when monitoring is not implemented, so that this can be used to report built-in monitoring events.
N08 Periodic Event	see N07.

5. Information Model vs. Device Model

As described above, the terms Information Model and Device Model refer to different concepts. The Information Model refers to a model of the information structure upon which the messages and datatypes in OCPP are based, whereas the Device Model refers to a generalized mechanism within OCPP to enable any model of Charging Station to report how it is build up so, it can be managed from any CSMS without defining the structure of the Charging Station in advance.

The messages that are used for Device Management are therefore part of the Information Model and the objects that are used for modelling a device ('Component' and 'Variable') are also part of the Information Model.

6. Using OCPP for other purposes than EV charging

As indicated in the introduction of this document, OCPP is primarily intended for two way communication between a CSMS and a Charging Station. However, with the addition of the Device Model as described in the chapter [Device Model](#), OCPP can additionally be used for other purposes. For example, the reporting of Events or Status changes in transformers or stand-alone battery packs might also be useful for companies that are rolling out EV charging infrastructure. In this example, a BootNotification could be used to connect these devices to a management system. In the device model a device that is not a Charging Station, can be recognized by the fact that the component Charging Station is not present at the top level. At the moment the OCPP specification does not provide use cases for non Charging Station devices. However, they may be added in a future version of OCPP.

7. Numbering

This section is normative.

7.1. EVSE numbering

To enable the CSMS to address all the EVSEs of a Charging Station, EVSEs MUST always be numbered in the same way.

EVSEs numbering (evselds) MUST be as follows:

- The EVSEs MUST be sequentially numbered, starting from 1 at every Charging Station (no numbers may be skipped).
- evselds MUST never be higher than the total number of EVSEs of a Charging Station
- For operations initiated by the CSMS, evseld 0 is reserved for addressing the entire Charging Station.
- For operations initiated by the Charging Station (when reporting), evseld 0 is reserved for the Charging Station main controller.

Example: A Charging Station with 3 EVSEs: All EVSEs MUST be numbered with the IDs: 1, 2 and 3. It is advisable to number the EVSEs of a Charging Station in a logical way: from left to right, top to bottom incrementing.

7.2. Connector numbering

To enable the CSMS to address all the Connectors of a Charging Station, Connectors MUST always be numbered in the same way.

Connector numbering (connectorIds) MUST be as follows:

- The connectors are numbered (increasing) starting at connectorId 1 on every EVSE
- Every connector per EVSE has a unique number
- ID of the first Connector of an EVSE MUST be 1
- Additional Connectors of the same EVSE MUST be sequentially numbered (no numbers may be skipped)
- connectorIds MUST never be higher than the total number of connectors on that EVSE

Example: A Charging Station with 3 EVSEs that each have 2 connectors, is numbered as follows:

- EVSE 1 has connectors with connectorId 1 and 2
- EVSE 2 has connectors with connectorId 1 and 2
- EVSE 3 has connectors with connectorId 1 and 2

7.3. Transaction IDs

TransactionIds are now generated by the Charging Station and MUST be unique on this Charging Station for every started transaction.

In OCPP 1.x this was done by the CSMS.

The format of the transaction ID is left to implementation. This MAY for example be an incremental number or an UUID.

8. Topologies supported by OCPP

This chapter shows a number of topologies for using OCPP. As indicated in the introduction, OCPP was originally used for a setup where each Charging Station communicates directly with the CSMS. It is important to keep in mind that OCPP has no knowledge of the topology of the Charging Station network. The following figure shows the possible components in a setup using OCPP and the relations between these components:

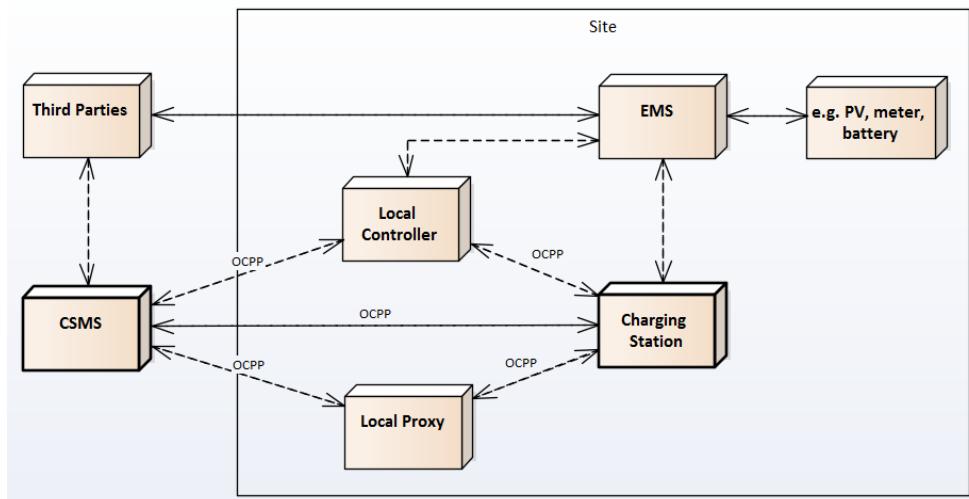


Figure 6. Possible components in a setup using OCPP

8.1. Charging Station(s) directly connected to CSMS

Description

This is the basic setup for using OCPP.

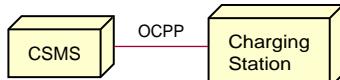


Figure 7. Charging Station directly connected to CSMS

8.2. Multiple Charging Stations connected to CSMS via Local Proxy

Description

In some situations it is desirable to route all communications for a group of Charging Stations through a single network node (i.e. modem, router, etc.). A typical example is the situation where a number of Charging Stations are located in an underground parking garage with little or no access to the mobile network. In order to provide access to mobile data the Charging Stations are linked to a central data communications unit over a LAN. This central unit connects to the mobile network and acts as a proxy between CSMS and Charging Stations. Such a unit is called a "local proxy" (LP) in OCPP. A local proxy acts as a message router. Neither the CSMS nor the Charging Stations are aware of the topology of the network. For the Charging Stations in the group the local proxy "is" the CSMS. Similarly, for the CSMS the local proxy "is" the Charging Station. The diagram below illustrates this configuration.

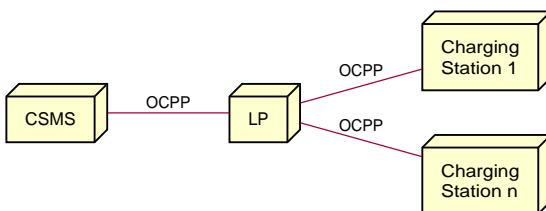


Figure 8. Multiple Charging Stations connected to CSMS via Local Proxy

8.3. Multiple Charging Stations connected to CSMS via Local Controller

Description

Whereas a [local proxy](#) does little more than route OCPP messages, a Local Controller can send messages to its Charging Stations, independently of the CSMS. A typical usage for this is the local smart charging case described in the Smart Charging chapter of Part 2 of OCPP, where a Local Controller can impose charge limits on its Charging Stations. In order for a Local Controller to be addressed by the CSMS, it needs to have its own Charging Station identity. From the point of view from OCPP, the Local Controller will just be a Charging Station (without any EVSEs/Connectors). The CSMS will possess the logic to deal with the Local Controller in order to support, for example, local smart charging. It is up to the implementation of the CSMS, whether the group topology is manually configured or deduced from the network based on IP addresses and information in BootNotifications. The diagram below illustrate this configuration.

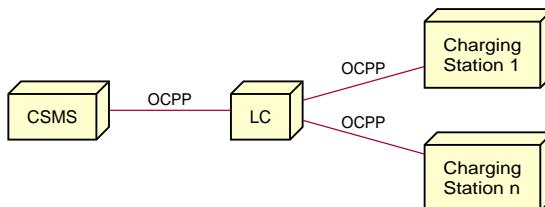


Figure 9. Multiple Charging Stations connected to CSMS via Local Controller

NOTE

Technically this topology can be realized in multiple ways. When using this setup with websockets, this implies that when a Charging Station connects to the Local Controller, it should open a websocket connection with the same address to the CSMS. The advantages of this approach is that the Local Controller can see all the messages and act on it, messages don't have to wait, firmware updates etc. on the Charging Stations are possible and the CSMS does not need special software. It could (in big installations) lead to a lot of websocket connections between CSMS and LC needed. For further information, please refer to OCPP implementation guide in Part 4.

8.4. Non-OCPP Charging Stations connected to CSMS via OCPP Local Controller

Description

This setup has multiple non-OCPP Charging Stations that are abstracted away using a OCPP enabled Local Controller. When applying OCPP in this situation, the LC should be considered as a Charging Station with many EVSEs or the LC should act as multiple OCPP Charging Stations (having their own Charging Station Identity).

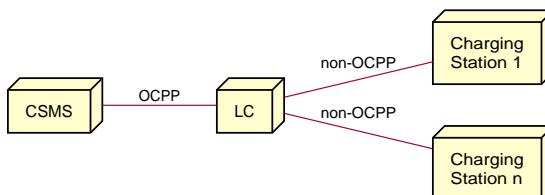


Figure 10. Multiple non-OCPP Charging Stations connected to CSMS via Local Controller

8.5. DSO control signals to CSMS

Description

This is a set up in which the CSMS is the only application sending signals to its Charging Stations, but the CSMS receives smart charging signals from a DSO based on (most likely) grid constraints. This means that a non-OCPP signal such as OpenADR or OSCP is received and based on this signal, the CSMS limits charging on its Charging Stations. CSOs that want full control over their Charging Station use this architecture, this way they are in control of the amount of energy being used by their Charging Stations. This can be done by sending charging profiles / charging schedules to Charging Stations.



Figure 11. Smart Charging - DSO control signals to CSMS

8.6. Parallel control by CSMS and EMS

Description

In a (semi-)private situation where a Charging Station is not only connected to the CSMS, but also to an Energy Management System, some form of parallel control should be supported. OCPP should at least be used for Charging Station maintenance, but OCPP 2.0.1 also supports reporting external smart charging control limits. So if the Energy Management System decides that charging at a later time is "better", the Energy Management System can impose an external limit (e.g. 0) to a Charging Station, which the Charging Station in turn can report to the CSMS via OCPP. The Energy Management System might get input from e.g. Local port of Smart Meter to prevent overloading connection but can also have other reasons for not charging (e.g. weather conditions).



Figure 12. Parallel control by CSMS and EMS

9. Part 1 Appendix: OCPP Information Model

9.1. Explanation of UML representation and message generation

In the next paragraph, the UML schemes of the OCPP Information Model are shown. The model is based on the Common Information Model (CIM) and to some extent to the CEFACt naming standards (only part of the standard). The objects in the model are named *BusinessComponents* and inherit properties from the CIM *IdentifiedObject*, such as MRID and Name. In the UML diagrams the attributes that are inherited from *IdentifiedObject* are shown under the *IdentifiedObject* stereotype (between < >). Other attributes are listed under the stereotype <> Content >.

The messages in OCPP are derived from the model represented in the next paragraph, in a 3 step process:

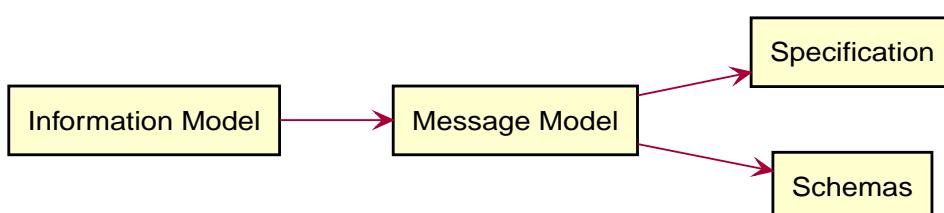


Figure 13. Process from information Model to Messages / schemas

After creating the Information Model, the messages are created based on the Information Model. However, in this transition (first arrow), some rules are (manually) applied for modelling messages. The most important rule that is applied, is that messages containing a reference to a <class> with only one <field>, are replaced by a field with the name <class><field>. For example, if a message contains a Transaction, with only an Id, this is replaced by a transactionId.

In the next step, when generating the messages and datatypes section of Part 2 of the specification, for readability, all Core DataTypes such as *CounterType*, are replaced by the Primitive DataType they refer to (except for enumerations) in this example *integer*.

9.2. Visual Representation of OCPP Information Model

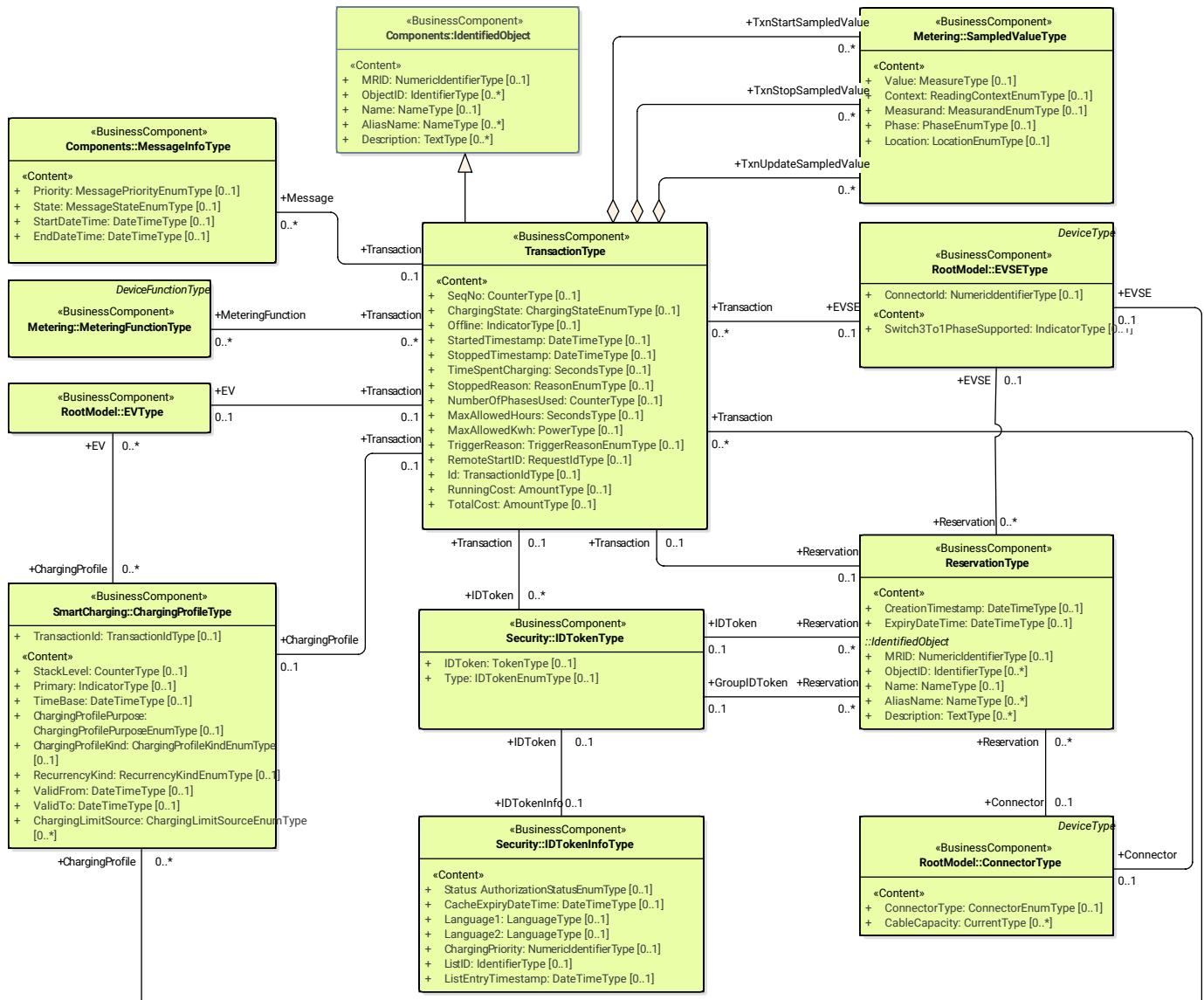


Figure 14. OCPP Information Model: Transactions

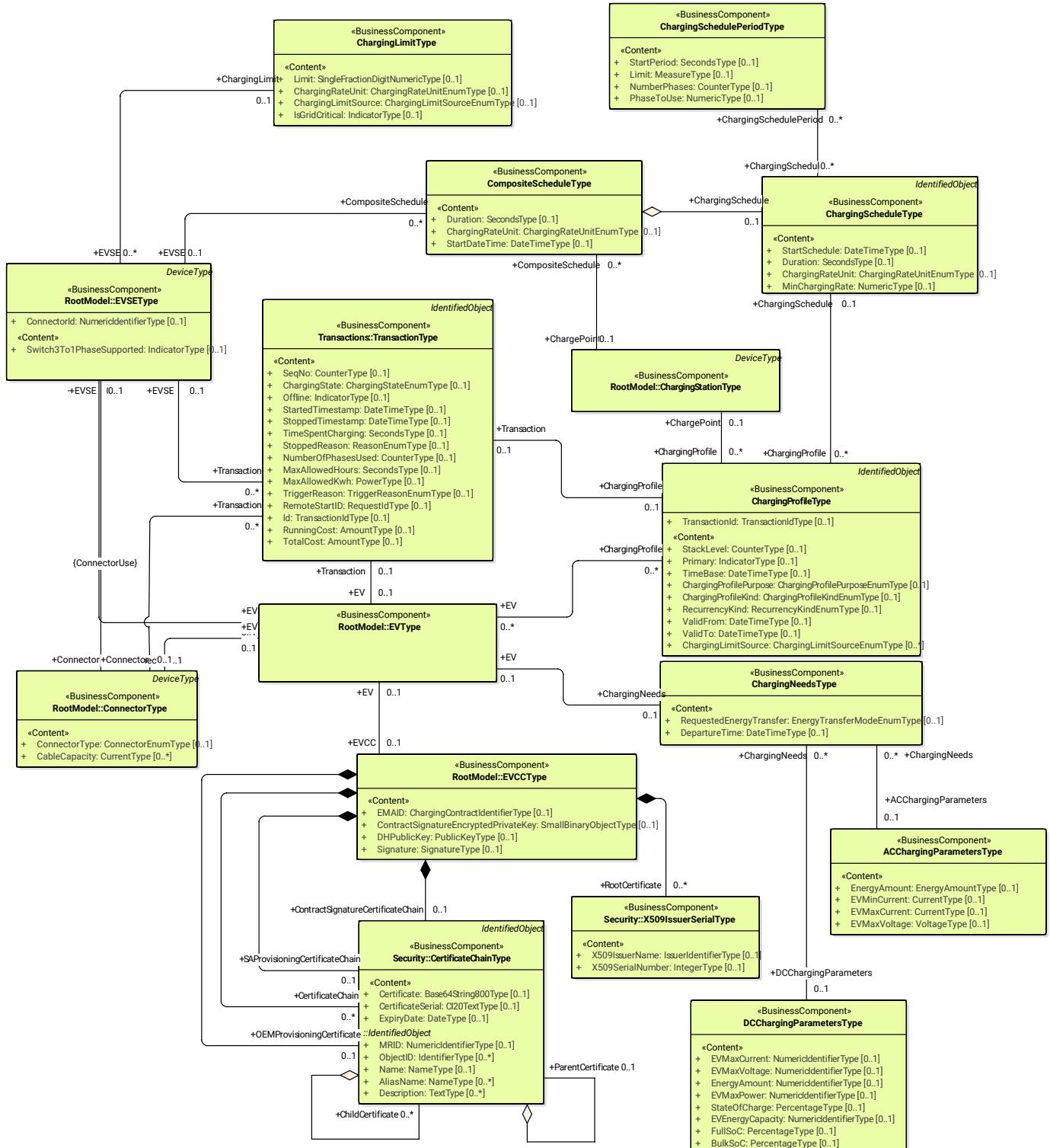


Figure 15. OCPP Information Model: SmartCharging

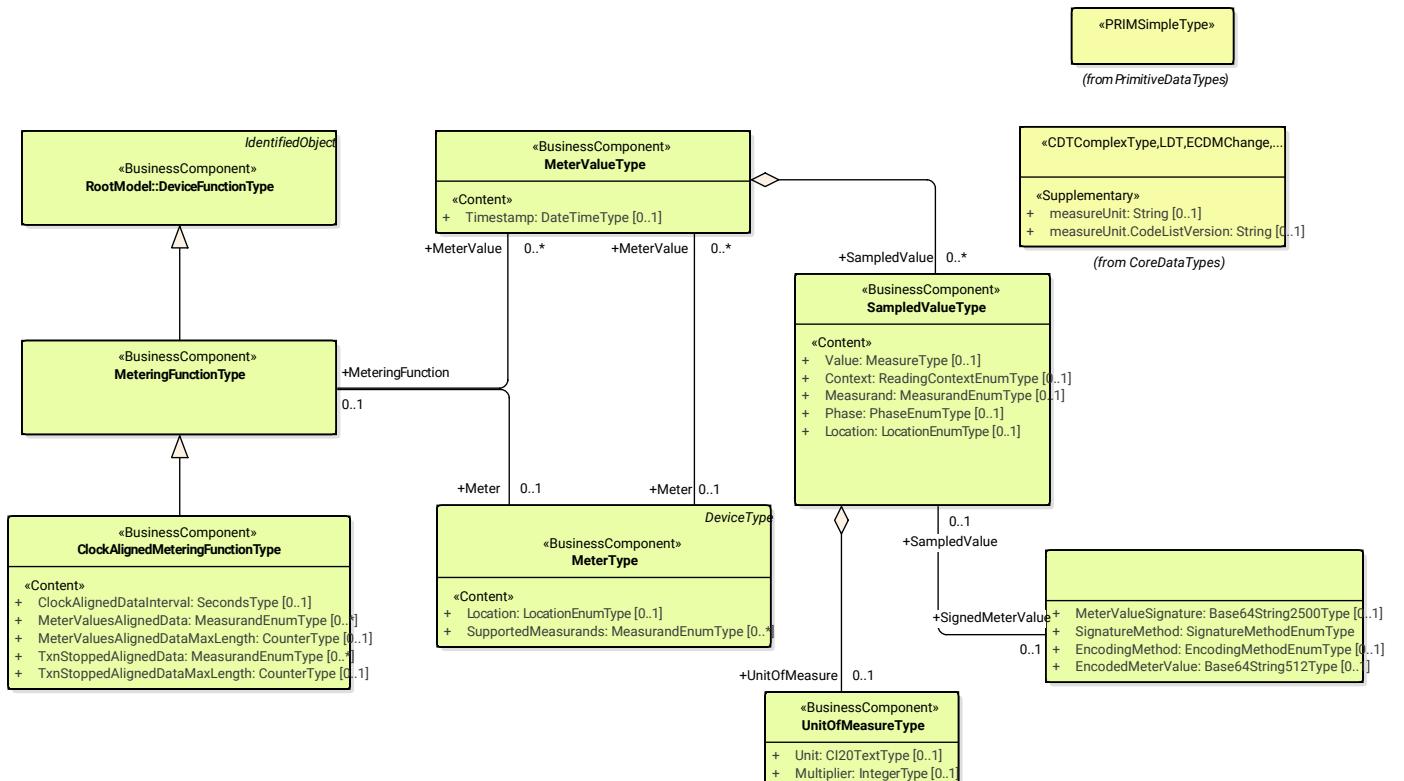


Figure 16. OCPP Information Model: Metering

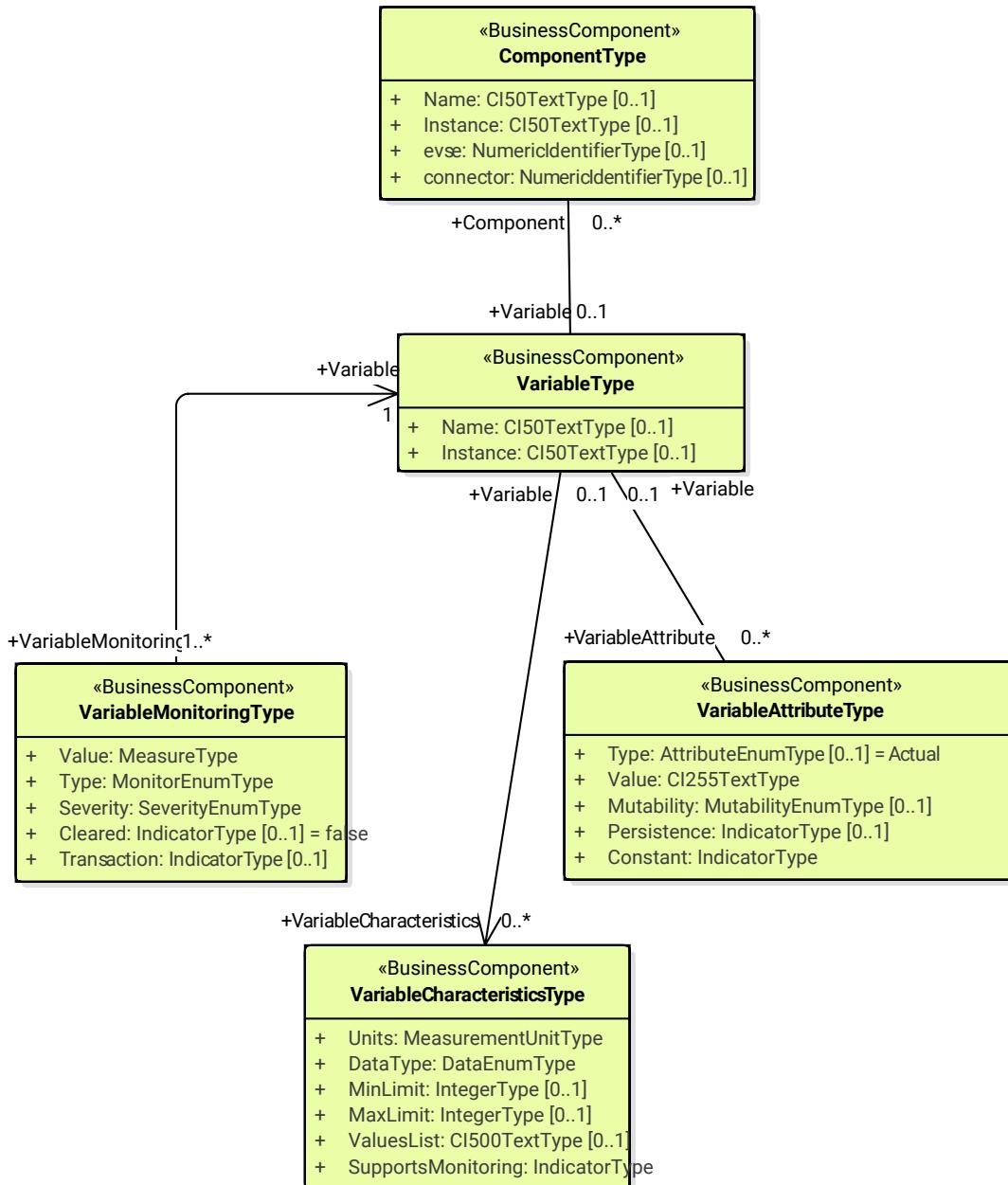


Figure 17. OCPP Information Model: Device Model

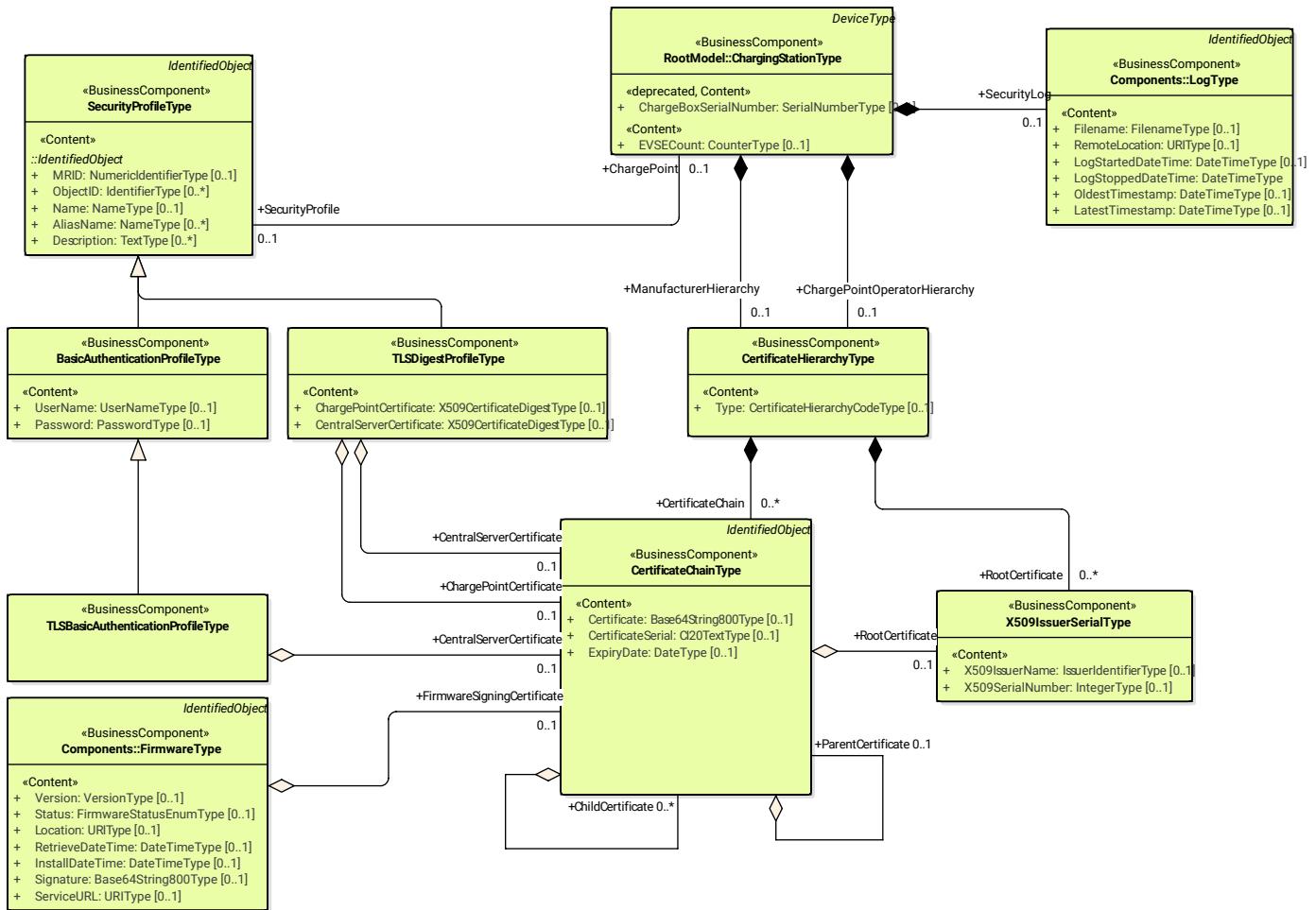


Figure 18. OCPP Information Model: Security-Profiles

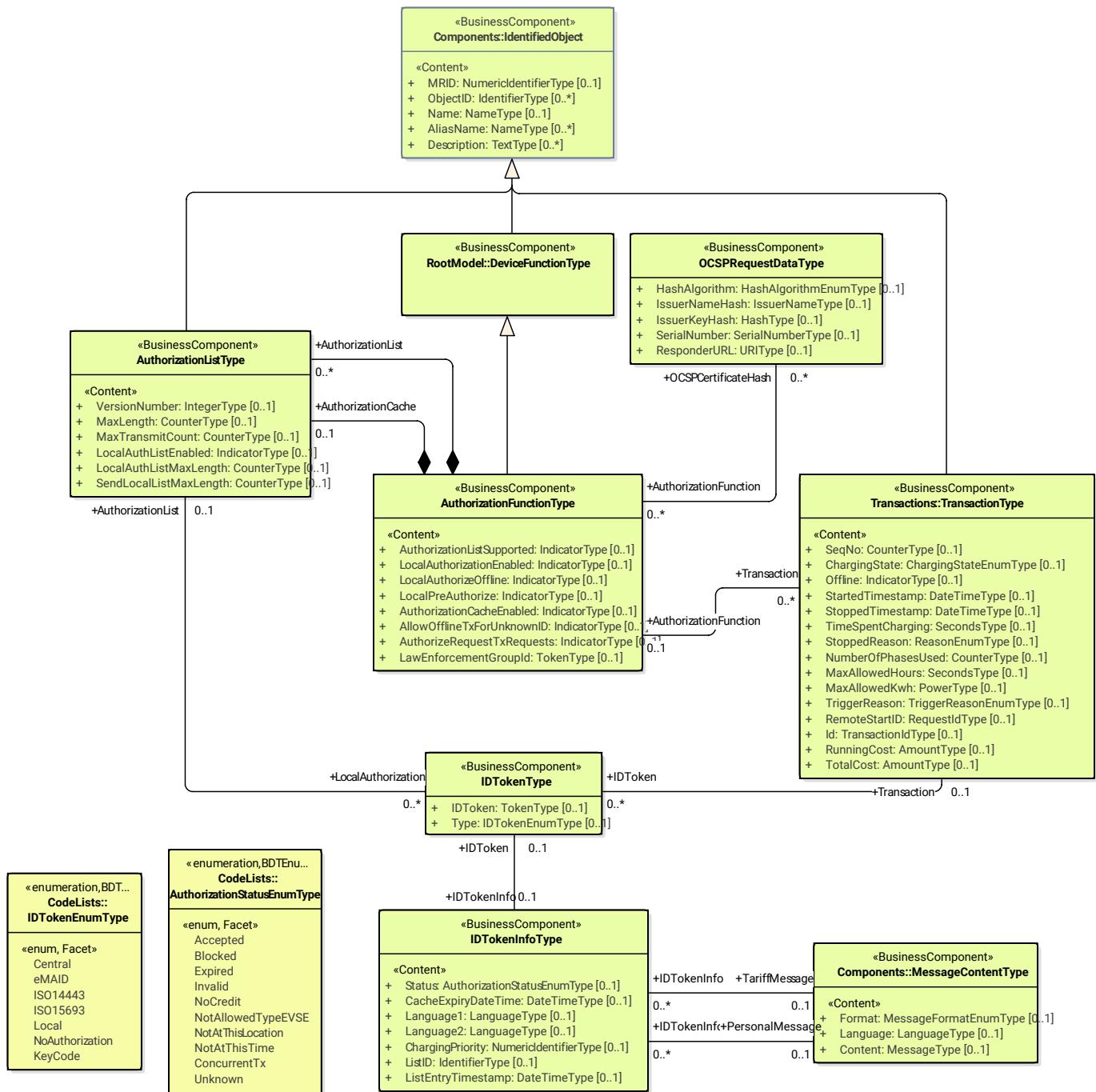


Figure 19. OCPP Information Model: Security-Authorization

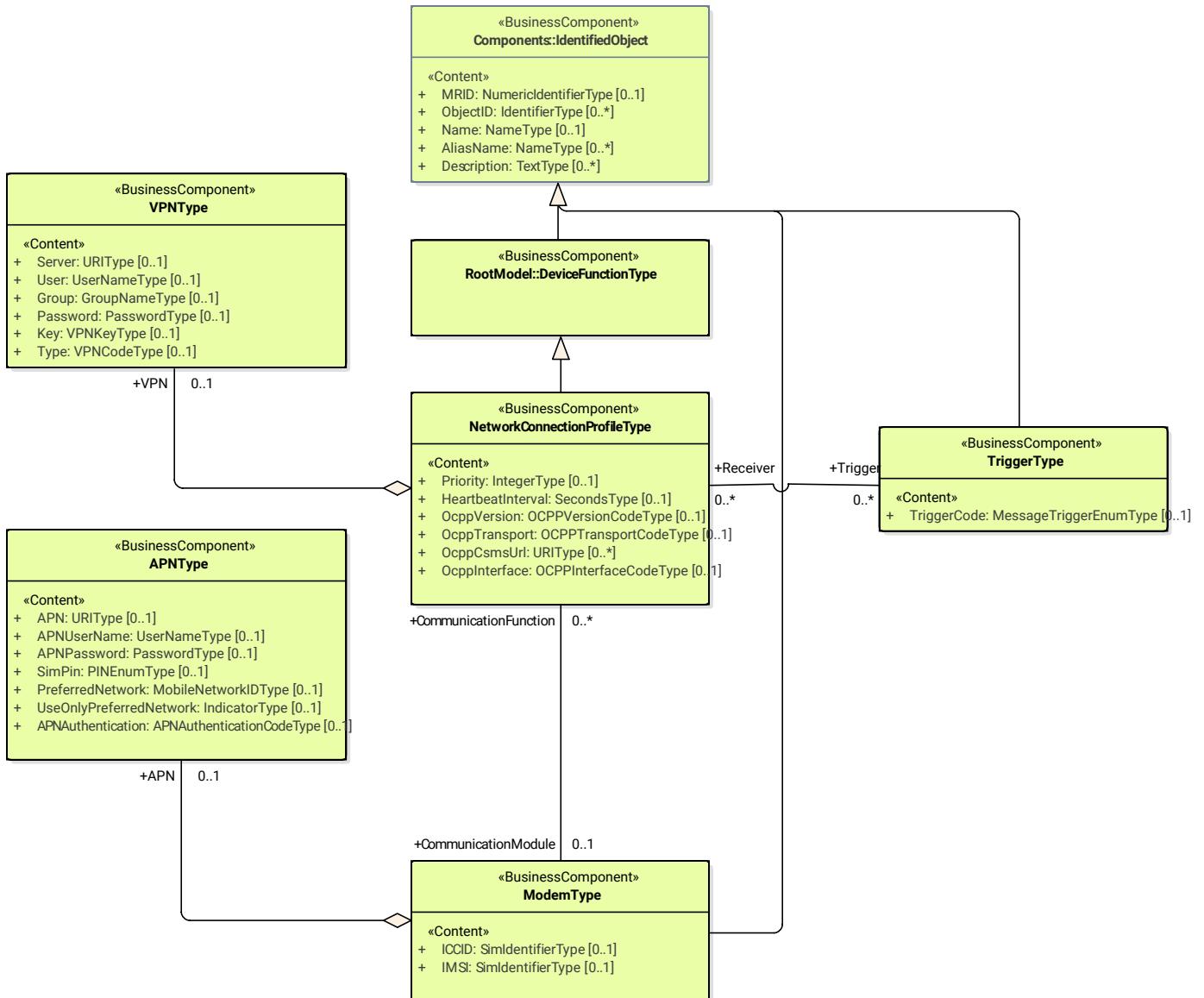


Figure 20. OCPP Information Model: Communication

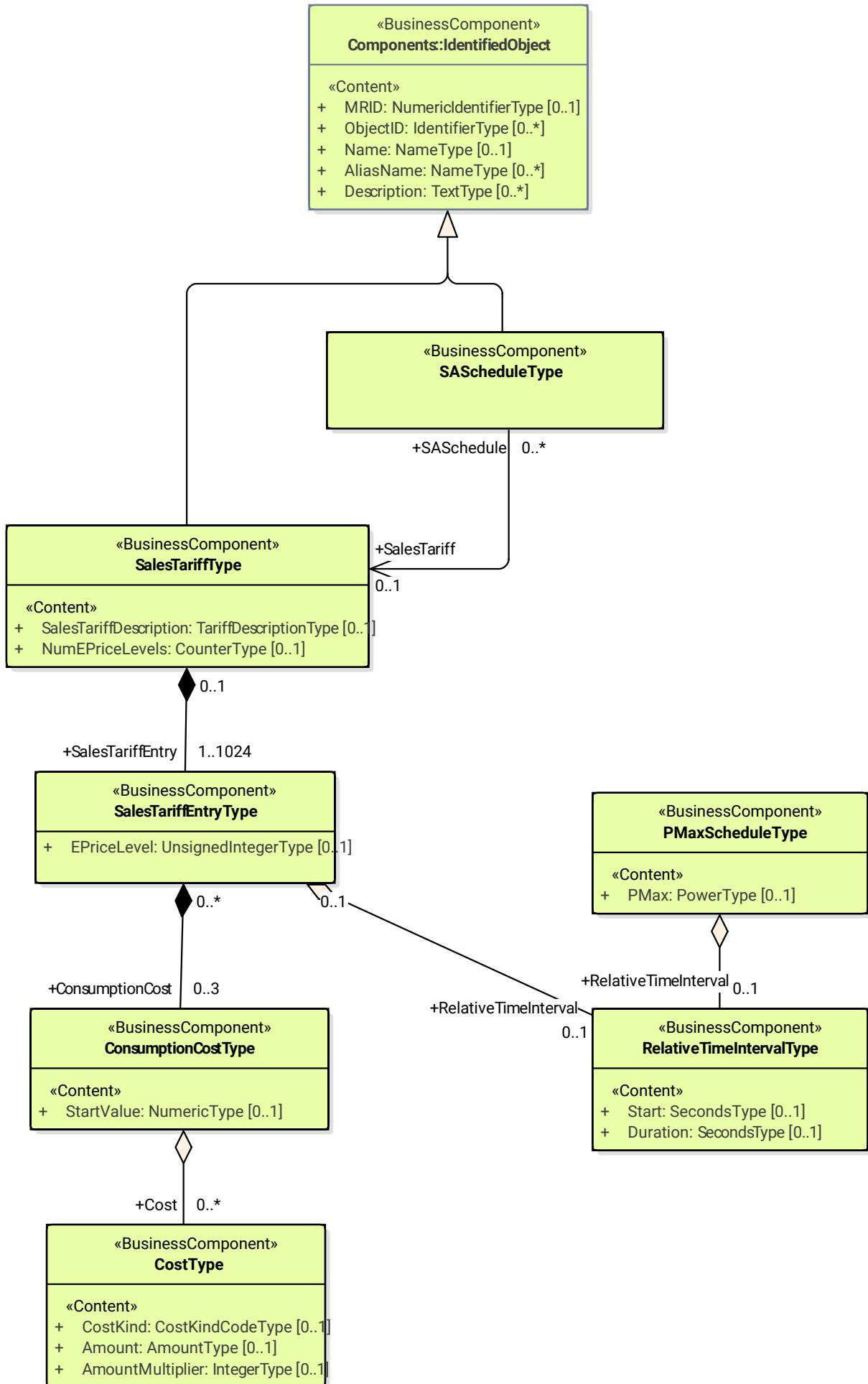


Figure 21. OCPP Information Model: SecondaryActorSchedule



OCPP 2.0.1

Part 2 - Appendices

v1.3 FINAL, 2022-12-15

Table of Contents

Disclaimer	1
Version History	2
1. Security Events	3
2. Standardized Units of Measure	4
3. Standardized Components	5
3.1. Controller Components	5
3.2. Physical Components	13
3.3. Summary List of Standardized Components	29
4. Standardized Variables	32
5. Reason Codes	36

Disclaimer

Copyright © 2010 – 2022 Open Charge Alliance. All rights reserved.

This document is made available under the **Creative Commons Attribution-NoDerivatives 4.0 International Public License** (<https://creativecommons.org/licenses/by-nd/4.0/legalcode>).

Version History

Appendix version	Date	OCPP Version	Description
1.3	2022-12-15	OCPP 2.0.1	Appendix version for Errata 2 (2022) Updated parts are marked with " <i>(Updated in v1.3)</i> ".
1.2	2021-10-01	OCPP 2.0.1	Appendix version for Errata 1 (2021) Appendix 3: Updated components are marked with " <i>(Updated in v1.2)</i> ". Appendix 3: Added ConnectedEV component for info from ISO15118 and CHAdeMO. Appendix 5: Added reason MissingDeviceModelInfo
1.1	2020-03-23	OCPP 2.0.1	Update for OCPP 2.0.1
1.0	2018-04-11	OCPP 2.0	First release of this Appendix for OCPP 2.0

Appendix 1. Security Events

The table below provides a list of security events. Security events that are implemented SHALL be stored at the security log and security events that are implemented and marked as critical SHALL also be pushed to the CSMS.

This is a non-exhaustive list of security events, when a security event matches the *description* of one of the Security Events in this section, for interoperability reasons, the Security Event from this section SHALL be used, instead of adding a new (proprietary) Security Event. Some security events like; *InvalidCsmsCertificate*, *InvalidChargingStationCertificate*, etc. are mandatory to be implemented. Please refer to Part 2 - Specification for which security events are mandatory to be implemented.

(Updated in v1.3)

Security Event	Description	Critical
FirmwareUpdated	The Charging Station firmware is updated	Yes
FailedToAuthenticateAtCsms	The authentication credentials provided by the Charging Station were rejected by the CSMS	No
CsmsFailedToAuthenticate	The authentication credentials provided by the CSMS were rejected by the Charging Station	No
SettingSystemTime	The system time on the Charging Station was changed more than <code>ClockCtrlr.TimeAdjustmentReportingThreshold</code> seconds	Yes
StartupOfTheDevice	The Charging Station has booted	Yes
ResetOrReboot	The Charging Station was rebooted or reset	Yes
SecurityLogWasCleared	The security log was cleared	Yes
ReconfigurationOfSecurityParameters	Security parameters, such as keys or the security profile used, were changed	No
MemoryExhaustion	The Flash or RAM memory of the Charging Station is getting full	Yes
InvalidMessages	The Charging Station has received messages that are not valid OCPP messages, if signed messages, signage invalid/incorrect	No
AttemptedReplayAttacks	The Charging Station has received a replayed message (other than the CSMS trying to resend a message because it there was for example a network problem)	No
TamperDetectionActivated	The physical tamper detection sensor was triggered	Yes
InvalidFirmwareSignature	The firmware signature is not valid	No
InvalidFirmwareSigningCertificate	The certificate used to verify the firmware signature is not valid	No
InvalidCsmsCertificate	The certificate that the CSMS uses was not valid or could not be verified	Yes
InvalidChargingStationCertificate	The certificate sent to the Charging Station using the <code>CertificateSignedRequest</code> message is not a valid certificate	Yes
InvalidTLSVersion	The TLS version used by the CSMS is lower than 1.2 and is not allowed by the security specification	Yes
InvalidTLSCipherSuite	The CSMS did only allow connections using TLS cipher suites that are not allowed by the security specification	Yes
MaintenanceLoginAccepted	Successful login to the local maintenance interface. It is recommended to include information like the user identification and the origin of the login attempt, which can be an ip-address or a touch screen for example, to the <code>techInfo</code> field. For this the following format is strongly recommended: <code>{"user": "...", "origin": "..."}</code>	Yes
MaintenanceLoginFailed	Failed login attempt to the local maintenance interface. It is recommended to include information like the user identification and the origin of the login attempt, which can be an ip-address or a touch screen for example, to the <code>techInfo</code> field. For this the following format is strongly recommended: <code>{"user": "...", "origin": "..."}</code>	Yes

Appendix 2. Standardized Units of Measure

The standardized values for Unit of Measure. Default value of "unit" is always "Wh".

Value	Description
A	Amperes (current)
ASU	Arbitrary Strength Unit (Signal Strength)
B	Bytes
Celsius	Degrees (temperature)
dB	Decibel (for example Signal Strength)
dBm	Power relative to 1mW ($10\log(P/1\text{mW})$)
Deg	Degrees (angle/rotation)
Fahrenheit	Degrees (temperature)
Hz	Hertz (frequency)
K	Degrees Kelvin (temperature)
lx	Lux (Light Intensity)
m	Meter (length)
ms2	m/s^2 (Acceleration)
N	Newton (Force)
Ohm	Ohm (Impedance)
kPa	kiloPascal (Pressure)
Percent	Percentage
RH	Relative Humidity%
RPM	Revolutions per Minute
s	Seconds (Time)
V	Voltage (DC or r.m.s. AC)
VA	Volt-Ampere (apparent power)
kVA	kiloVolt-Ampere (apparent power)
VAh	Volt-Ampere-hours (apparent energy)
kVAh	kiloVolt-Ampere-hours (apparent energy)
var	vars (reactive power)
kvar	kilovars (reactive power)
varh	var-hours (reactive energy)
kvarh	kilovar-hours (reactive energy)
W	Watts (power)
kW	kilowatts (power)
Wh	Watt-hours (energy). Default
kWh	kilowatt-hours (energy)

Appendix 3. Standardized Components

This appendix provides a list of all standardized component names for OCPP 2.0.1 for controller components and for physical components. A summary table listing just all components without variables is provided at the end of this appendix in [Summary List of Standardized Components](#).

3.1. Controller Components

This is the list of Standardized Controller Components for OCPP 2.0.1. and typical Variables that might be associated with them.

IMPORTANT

This list does not imply that these Components are required, nor does it imply that the listed Variables are required for a Component or no other Variables are allowed to be associated with a Component.

3.1.1. AlignedDataCtrlr

Description		
Logical Component responsible for configuration relating to the reporting of clock-aligned meter data.		
Variables	Type	Description
Enabled	boolean	If this variable reports a value of true, Aligned Data is enabled.
Available	boolean	If this variable reports a value of true, Aligned Data is supported.
Interval	integer	Size (in seconds) of the clock-aligned data interval, intended to be transmitted in the MeterValuesRequest message.
Measurands	MemberList	Clock-aligned measurand(s) to be included in MeterValuesRequest, every AlignedDataInterval seconds.
SendDuringIdle	boolean	If set to true, the Charging Station SHALL not send clock aligned meter values when a transaction is ongoing.
SignReadings	boolean	If set to true, the Charging Station SHALL include signed meter values in the TransactionEventRequest to the CSMS.
TxEndedInterval	integer	Size (in seconds) of the clock-aligned data interval, intended to be transmitted in the TransactionEventRequest (eventType = Ended) message.
TxEndedMeasurands	MemberList	Clock-aligned periodic measurand(s) to be included in the meterValues element of TransactionEventRequest (eventType = Ended) for every TxEndedAlignedDataInterval of the transaction.

3.1.2. AuthCtrlr (Updated in v1.2)

Description		
Logical Component responsible for configuration relating to the use of authorization for Charging Station use.		
Variables	Type	Description
Enabled	boolean	If set to <i>false</i> , then no authorization is done before starting a transaction or when reading an idToken. If an idToken was provided, then it will be put in the <i>idToken</i> field of the TransactionEventRequest. If no idToken was provided, then <i>idToken</i> in TransactionEventRequest will be left empty and type is set to NoAuthorization.
AdditionalInfoItemsPerMessage	integer	Maximum number of AdditionalInfo items that can be sent in one message.
AuthorizeRemoteStart	boolean	Whether a remote request to start a transaction in the form of RequestStartTransactionRequest message should be authorized beforehand like a local action to start a transaction.
DisableRemoteAuthorization	boolean	When set to <i>true</i> this instructs the Charging Station to not issue any AuthorizationRequests, but only use Authorization Cache and Local Authorization List to determine validity of idTokens.
LocalAuthorizeOffline	boolean	Whether the Charging Station, when Offline, will start a transaction for locally-authorized identifiers.
LocalPreAuthorize	boolean	Whether the Charging Station, when online, will start a transaction for locally-authorized identifiers without waiting for or requesting an AuthorizeResponse from the CSMS.

Description		
MasterPassGroupId	string	IdTokens that have this id as groupId belong to the Master Pass Group.
OfflineTxForUnknownIdEnabled	boolean	If this key exists, the Charging Station supports Unknown Offline Authorization.

3.1.3. AuthCacheCtrlr (*Updated in v1.2*)

Description		
Logical Component responsible for configuration relating to the use of a local cache for authorization for Charging Station use.		
Variables	Type	Description
Enabled	boolean	If this variable exists, the Charging Station supports an Authorization Cache.
Available	boolean	If this variable reports a value of true, Authorization Cache is supported.
LifeTime	integer	Indicates in seconds how long it takes until a token expires in the authorization cache since it is last used.
Policy	OptionList	Cache Entry Replacement Policy: (LRU,LFU) LeastRecentlyUsed or LeastFrequentlyUsed. Allowed values: LRU, LFU.
DisablePostAuthorize	boolean	When set to <i>true</i> this variable disables the behavior to request authorization for an idToken that is stored in the cache with a status other than Accepted, as stated in C10.FR.03 and C12.FR.05.

3.1.4. CHAdemoCtrlr (*Updated in v1.2*)

Description		
A CHAdemo Controller component communicates with an EV using the wired CANbus protocol to exchange information and control charging using the CHAdemo protocol		
Variables	Type	Description
Enabled	boolean	CHAdemo controller enabled
Active	boolean	Connected
Complete	boolean	Protocol session ended normally
Tripped	boolean	CHAdemo protocol terminated abnormally
Problem	boolean	CHAdemo controller fault
SelftestActive(Set)	boolean	Start self-test by setting to true
SelftestActive	boolean	Self-test running when reported as true
Specific CHAdemo interface data from vehicle:		
CHAdemoProtocolNumber	integer	CHAdemo protocol number (H'102.0)
VehicleStatus	boolean	Vehicle status (H'102.5.3)
DynamicControl	boolean	Vehicle is compatible with dynamic control (H'110.0.0)
HighCurrentControl	boolean	Vehicle is compatible with high current control (H'110.0.1)
HighVoltageControl	boolean	Vehicle is compatible with high voltage control (H'110.1.2)
AutoManufacturerCode	integer	Auto manufacturer code (H'700.0) <i>A single byte manufacturer code assigned by CHAdemo association</i>

3.1.5. ClockCtrlr

Description		
Provides a means to configure management of time tracking by Charging Station.		
Variables	Type	Description
DateTime	dateTime	Contains the current date and time.
NtpServerUri	string	This contains the address of the NTP server. Multiple NTP servers can be configured as backups, etc. If the NTP client supports it, it can also connect to multiple NTP servers simultaneous to get a more reliable time source. Variable instance value is single digit NTP priority (1=highest).

Description		
NtpSource	string	When an NTP client is implemented, this variable can be used to configure the client: Use the NTP server provided via DHCP, or use the manually configured NTP server.
TimeOffset	string	Configured local time offset in the format: "+01:00", "-02:00" etc.
NextTimeOffsetTransitionDateTime	dateTime	Date time of the next time offset transition.
TimeSource	string	Via this variable, the Charging Station provides the CSMS with the option to configure a clock source, if more than 1 are implemented.
TimeZone	string	Configured current local time zone in the format: "Europe/Oslo", "Asia/Singapore" etc.
TimeAdjustmentReportingThreshold	integer	If set, then time adjustments with an absolute value in seconds larger than this need to be reported as a security event SettingSystemTime.

3.1.6. CustomizationCtrlr (New in v1.2)

Description		
Logical Component responsible for configuration relating to custom vendor-specific implementations, using the DataTransfer message and CustomData extensions.		
Variables	Type	Description
CustomImplementationEnabled	boolean	This standard configuration variable can be used to enable/disable custom implementations that the Charging Station supports. The instance name of the variable matches the <i>vendorId</i> of the customization in CustomData or DataTransfer messages.

3.1.7. DeviceDataCtrlr

Description		
Logical Component responsible for configuration relating to the exchange and storage of Charging Station Device Model data.		
Variables	Type	Description
BytesPerMessage	integer	Message Size (in bytes) - maxLimit used to report constraint on message size. Which message is specified in the instance.
ItemsPerMessage	integer	Maximum number of entries that can be sent in one message. Which entries in which message is specified in the instance.
ValueSize	integer	Can be used to limit the following fields: SetVariableData.attributeValue, GetVariableResult.attributeValue, VariableAttribute.value, VariableCharacteristics.valueList and EventData.actualValue.

3.1.8. DisplayMessageCtrlr

Description		
Logical Component responsible for configuration relating to the display of messages to Charging Station users.		
Variables	Type	Description
Enabled	boolean	Whether Display Message is enabled.
Available	boolean	Whether Display Message is supported.
DisplayMessages	integer	Amount of different messages that are currently configured in this Charging Station, via SetDisplayMessageRequest.
PersonalMessageSize	integer	Max size (in characters) of the personal message element of the IdTokenInfo data (0 specifies no personal data may be stored).
SupportedFormats	MemberList	List of message formats supported by this Charging Station. Possible values: See MessageFormatEnumType.

3.1.9. ISO15118Ctrlr (Updated in v1.3)

Description		
Communicates with an EV to exchange information and control charging using the ISO 15118 protocol.		
Variables	Type	Description
Enabled	boolean	ISO15118 controller enabled
Active	boolean	Connected
Tripped	boolean	ISO15118 communication session aborted
Complete	boolean	ISO15118 communication session ended
Problem	boolean	ISO15118 controller fault
SeccId	string	The name of the SECC in the string format as required by ISO 15118.
SelftestActive(Set)	boolean	Start self-test by setting to true
SelftestActive	boolean	Self-test running when reported as true
ContractValidationOffline	boolean	Supports validation of a contract certificate when offline
CentralContractValidationAllowed	boolean	Contract certificates can be validated by the CSMS
PnCEnabled	boolean	If this variable is <i>true</i> , then ISO 15118 plug and charge as described by use case C07 - Authorization using Contract Certificates is enabled. If this variable is <i>false</i> , then ISO 15118 plug and charge as described by use case C07 - Authorization using Contract Certificates is disabled.
V2GCertificateInstallationEnabled	boolean	If this variable is <i>true</i> , then ISO 15118 V2G Charging Station certificate installation as described by use case A02 - Update Charging Station Certificate by request of CSMS and A03 - Update Charging Station Certificate initiated by the Charging Station is enabled. If this variable is <i>false</i> , then ISO 15118 V2G Charging Station certificate installation as described by use case A02 - Update Charging Station Certificate by request of CSMS and A03 - Update Charging Station Certificate initiated by the Charging Station is disabled.
ContractCertificateInstallationEnabled	boolean	If this variable is <i>true</i> , then ISO 15118 contract certificate installation/update as described by use case M01 - Certificate installation EV and M02 - Certificate Update EV is enabled. If this variable is <i>false</i> , then ISO 15118 contract certificate installation/update as described by use case M01 - Certificate installation EV and M02 - Certificate Update EV is disabled.
RequestMeteringReceipt	boolean	If this variable is <i>true</i> , then Charging Station shall request a metering receipt from EV before sending a fiscal meter value to CSMS.
OrganizationName	string	The organizationName of the CSO operating the charging station. It is used as the organizationName (O) of the SECC leaf certificate. Example: "John Doe Charging Services Ltd" Note: This value will usually be identical to SecurityCtrlr.OrganizationName, but it does not have to be.
CountryName	string	The countryName of the SECC in the ISO 3166-1 format. It is used as the countryName (C) of the SECC leaf certificate. Example: "DE"
Specific ISO15118 interface data from vehicle:		
MaxScheduleEntries	integer	MaxEntriesSAScheduleType (15118-2) or MaximumSupportingPoints (15118-20) Number of allowed schedule periods
RequestedEnergyTransferMode	OptionList	RequestedEnergyTransferMode "AC_single_phase_core", "AC_three_phase_core", "DC_core", "DC_extended", "DC_combo_core", "DC_unique"

3.1.10. LocalAuthListCtrlr (*Updated in v1.2*)

Description		
Logical Component responsible for configuration relating to the use of Local Authorization Lists for Charging Station use.		
Variables	Type	Description
Enabled	boolean	Whether Local Authorization List is enabled.
Entries	integer	Amount of IdTokens currently in the Local Authorization List. The maxLimit of this variable SHALL be provided to report the maximum number of IdTokens that can be stored in the Local Authorization List.

Description		
Available	boolean	Whether Local Authorization List is supported.
ItemsPerMessage	integer	Maximum number of identifications that can be sent in a single SendLocalListRequest.
BytesPerMessage	integer	Message Size (in bytes) - puts a constraint on SendLocalListRequest message size.
Storage	integer	Indicates the number of bytes currently used by the Local Authorization List. MaxLimit indicates the maximum number of bytes that can be used by the Local Authorization List.
DisablePostAuthorize	boolean	When set to <i>true</i> this variable disables the behavior to request authorization for an idToken that is stored in the local authorization list with a status other than Accepted, as stated in C14.FR.03.

3.1.11. MonitoringCtrlr (*Updated in v1.3*)

Description		
Variables	Type	Description
Enabled	boolean	Whether Monitoring is enabled.
Available	boolean	Whether Monitoring is supported.
ItemsPerMessage	integer	Maximum number of items.
BytesPerMessage	integer	Message Size (in bytes) - puts constraint on message size.
MonitoringBase	optionList	Currently used MonitoringBase. (readonly)
MonitoringLevel	integer	Currently use MonitoringLevel (readonly)
OfflineQueuingSeverity	integer	When set and the Charging Station is offline, the Charging Station shall queue any notifyEventRequest messages triggered by a monitor with a severity number equal to or lower than the severity configured here. Value ranging from 0 (Emergency) to 9 (Debug).
ActiveMonitoringBase	OptionList	Shows the currently used MonitoringBase. Valid values according MonitoringBaseEnumType: All, FactoryDefault, HardwiredOnly. (readonly)
ActiveMonitoringLevel	integer	Shows the currently used MonitoringLevel. Valid values are severity levels of SetMonitoringLevelRequest: 0-9. (readonly)

3.1.12. OCPPCommCtrlr (*Updated in v1.3*)

Description		
Variables	Type	Description
ActiveNetworkProfile	string	Indicates the configuration profile the station uses at that moment to connect to the network.
FileTransferProtocols	MemberList	List of supported file transfer protocols.
HeartbeatInterval	integer	Interval in seconds of inactivity (no OCPP exchanges) with CSMS after which the Charging Station should send HeartbeatRequest.
MessageAttempts	integer	How often the Charging Station should try to submit a TransactionEventRequest message when the CSMS fails to process it.
MessageAttemptInterval	integer	How long in seconds the Charging Station should wait before resubmitting a TransactionEventRequest message that the CSMS failed to process.
MessageTimeout	integer	Message timeout in seconds. The message timeout setting in a Charging Station can be configured in the messageTimeout field in the NetworkConnectionProfile.
MinimumStatusDuration	integer	Minimum duration that a Charging Station or EVSE status is stable before StatusNotificationRequest is sent to the CSMS.
NetworkConfigurationPriority	string	A comma separated ordered list of the priority of the possible Network Connection Profiles.
NetworkProfileConnectionAttempts	integer	Specifies the number of connection attempts the Charging Station executes before switching to a different profile.

Description		
OfflineThreshold	integer	When the offline period in seconds of a Charging Station exceeds the OfflineThreshold it is recommended to send a StatusNotificationRequest for all its Connectors when the Charging Station is back online.
PublicKeyWithSignedMeterValue	boolean	This Configuration Variable can be used to configure whether a public key needs to be sent with a signed meter value.
QueueAllMessages	boolean	When this variable is set to true, the Charging Station will queue all message until they are delivered to the CSMS.
RetryBackOffRepeatTimes	integer	When the Charging Station is reconnecting, after a connection loss, it will use this variable for the amount of times it will double the previous back-off time.
RetryBackOffRandomRange	integer	When the Charging Station is reconnecting, after a connection loss, it will use this variable as the maximum value for the random part of the back-off time.
RetryBackOffWaitMinimum	integer	When the Charging Station is reconnecting, after a connection loss, it will use this variable as the minimum back-off time, the first time it tries to reconnect.
UnlockOnEVSideDisconnect	boolean	When set to true, the Charging Station SHALL unlock the cable on the Charging Station side when the cable is unplugged at the EV. For an EVSE with only fixed cables, the mutability SHALL be ReadOnly and the actual value SHALL be false. For a charging station with fixed cables and sockets, the variable is only applicable to the sockets.
WebSocketPingInterval	integer	Number of seconds between pings.
FieldLength	integer	This variable is used to report the length of <field> in <message> when it is larger than the length that is defined in the standard OCPP message schema.

3.1.13. ReservationCtrlr

Description		
Logical Component responsible for configuration relating to reservations.		
Variables	Type	Description
Enabled	boolean	Whether Reservation is enabled.
Available	boolean	Whether Reservation is supported.
NonEvseSpecific	boolean	If this configuration variable is present and set to true: Charging Station supports Reservation without specifying an EVSE.

3.1.14. SampledDataCtrlr

Description		
Logical Component responsible for configuration relating to the reporting of sampled meter data.		
Variables	Type	Description
Enabled	boolean	If this variable reports a value of true, Sampled Data is enabled.
Available	boolean	If this variable reports a value of true, Sampled Data is supported.
SignReadings	boolean	If set to true, the Charging Station includes signed meter values in the MeterValuesRequest to the CSMS.
TxEndedMeasurands	MemberList	Sampled measurands to be included in the meterValues element of TransactionEventRequest (eventType = Ended), every TxEndedSampleInterval seconds from the start of the transaction.
TxEndedInterval	integer	Interval in seconds between sampling of metering (or other) data, intended to be transmitted in the TransactionEventRequest (eventType = Ended) message.
TxStartedMeasurands	MemberList	Sampled measurand(s) to be taken at the start of any transaction to be included in the meterValues field of the first TransactionEventRequest message send at the start of a transaction (eventType = Started).
TxUpdatedMeasurands	MemberList	Sampled measurands to be included in the meterValues element of every TransactionEventRequest (eventType = Updated), every SampledDataTxUpdatedInterval seconds from the start of the transaction.
TxUpdatedInterval	integer	Interval in seconds between sampling of metering (or other) data, intended to be transmitted via TransactionEventRequest (eventType = Updated) messages.

Description		
RegisterValuesWithoutPhases	boolean	If this variable reports a value of <i>true</i> , then meter values of measurand Energy.Active.Import.Register will only report the total energy over all phases without reporting the individual phase values. If this variable is absent or <i>false</i> , then the value for each phase is reported, possibly also with a total value (depending on the meter).

3.1.15. SecurityCtrlr (*Updated in v1.3*)

Description		
Logical Component responsible for configuration relating to security of communications between Charging Station and CSMS.		
Variables	Type	Description
BasicAuthPassword	passwordString	The basic authentication password that is used for HTTP Basic Authentication. The datatype is passwordString consisting of minimum 16 and maximum 40 characters (alpha-numeric characters and the special characters allowed by passwordString). The password SHALL be sent as a UTF-8 encoded string (NOT encoded into octet string or base64). This configuration variable is write-only, so that it cannot be accidentally stored in plaintext by the CSMS when it reads out all configuration variables. This configuration variable is required unless only "security profile 3 - TLS with client side certificates" is implemented.
Identity	identifierString	The Charging Station identity. identity is an identifierString string, so it SHALL only contain characters that are allowed for identifierString. Maximum length was chosen to ensure compatibility with EVSE ID from [EMI3] "Part 2: business objects".
OrganizationName	string	This configuration variable is used to set the organization name of the CSO or an organization trusted by the CSO. It is used to set the O (organizationName) RDN in the subject field of the client certificate.
CertSigningWaitMinimum	integer	Seconds to wait before generating another CSR in case CSMS does not return a signed certificate.
CertSigningRepeatTimes	integer	Number of times to resend a SignCertificateRequest when CSMS does not return a signed certificate.

3.1.16. SmartChargingCtrlr

Description		
Logical Component responsible for configuration relating to smart charging.		
Variables	Type	Description
Enabled	boolean	Whether Smart Charging is enabled.
Available	boolean	Whether Smart Charging is supported.
ACPhaseSwitchingSupported	boolean	If defined and true, this EVSE supports the selection of which phase to use for 1 phase AC charging.
ProfileStackLevel	integer	Maximum acceptable value for stackLevel in a ChargingProfile. Since the lowest stackLevel is 0, this means that if SmartChargingCtrlr.ProfileStackLevel = 1, there can be at most 2 valid charging profiles per Charging Profile Purpose per EVSE.
RateUnit	MemberList	A list of supported quantities for use in a ChargingSchedule. Allowed values: 'A' and 'W'.
PeriodsPerSchedule	integer	Maximum number of periods that may be defined per ChargingSchedule.
ExternalControlSignalsEnabled	boolean	Indicates whether a Charging Station should respond to external control signals that influence charging.
NotifyChargingLimitWithSchedules	boolean	Indicates if the Charging Station should include the externally set charging limit/schedule in the message when it sends a NotifyChargingLimitRequest message. This might increase the data usage significantly, especially when an external system sends new profiles/limits with a short interval. Default is false when omitted.
Phases3to1	boolean	If defined and true, this Charging Station supports switching from 3 to 1 phase during a transaction.
Entries	integer	Amount of Charging profiles currently installed on the Charging Station. MaxLimit used to limit number of Charging profiles installed at any time.

Description		
LimitChangeSignificance	integer	If at the Charging Station side a change in the limit in a ChargingProfile is lower than this percentage, the Charging Station MAY skip sending a NotifyChargingLimitRequest or a TransactionEventRequest message to the CSMS. It is RECOMMENDED to set this key to a low value. See Smart Charging signals to a Charging Station from multiple actors.

3.1.17. TariffCostCtrlr

Description		
Variables	Type	Description
Enabled	boolean	Whether Tariff/cost is enabled.
Available	boolean	Whether Tariff/cost is supported.
TariffFallbackMessage	string	Message (and/or tariff information) to be shown to an EV Driver when there is no driver specific tariff information available.
TotalCostFallbackMessage	string	Message to be shown to an EV Driver when the Charging Station cannot retrieve the cost for a transaction at the end of the transaction.
Currency	string	Currency used by this Charging Station in a ISO 4217 formatted currency code.

3.1.18. TxCtrlr

Description		
Variables	Type	Description
EVConnectionTimeOut	integer	Interval in seconds from between "starting" of a transaction until incipient transaction is automatically canceled, due to failure of EV driver to (correctly) insert the charging cable connector(s) into the appropriate socket(s). The Charging Station SHALL go back to the original state, probably: 'Available'. "Starting" might be the swiping of the RFID, pressing a start button, a RequestStartTransactionRequest being received etc.
TxBeforeAcceptedEnabled	boolean	With this configuration variable the Charging Station can be configured to allow charging before having received a BootNotificationResponse with RegistrationStatus: Accepted. See: Transactions before being accepted by a CSMS.
TxStartPoint	MemberList	Defines when the Charging Station starts a new transaction: first transactioneventRequest: eventType = Started. When any event in the given list occurs, the Charging Station SHALL start a transaction. The Charging Station SHALL only send the Started event once for every transaction. It is advised to put all events that should be part of a transaction in the list, in case the start event never occurs. Because the possible events don't always have to come in the same order it is possible to provide a list of events. Which ever comes first will then cause a transaction to be started. For example: EVConnected, Authorized would mean that a transaction is started when an EV is detected (Cable is connected), or when an EV Driver swipes his RFID card en the CSMS successfully authorizes the ID for charging.
TxStopPoint	MemberList	Defines when the Charging Station ends a transaction: last transactioneventRequest: eventType = Ended. When any event in the given list is no longer valid, the Charging Station SHALL end the transaction. The Charging Station SHALL only send the Ended event once for every transaction.
MaxEnergyOnInvalidId	integer	Maximum amount of energy in Wh delivered when an identifier is deauthorized by the CSMS after start of a transaction.
StopTxOnInvalidId	boolean	whether the Charging Station will stop an ongoing transaction when it receives a non- Accepted authorization status in TransactionEventResponse for this transaction.
StopTxOnEVSideDisconnect	boolean	When set to true, the Charging Station SHALL administratively stop the transaction when the cable is unplugged from the EV.

3.2. Physical Components

This is a non-exhaustive list of Standardized Physical Components that SHALL be used when mapping a real Charging Station to the Device Model (for monitoring purposes).

When the physical component that is to be mapped, matches the *description* of one of the Standardized Components in this section, for interoperability reasons, the Standardized Component from this section SHALL be used, instead of adding a new (proprietary) component.

The list of typically used variables that is given for each Component is also non-exhaustive and all variables are optional. See also Part 1, paragraph 4.5. If a description of a variable is empty, please refer to the description in [Standardized Variables](#).

3.2.1. AccessBarrier

Description	
Allows physical access of vehicles to a charging site to be controlled.	
Typically used variables	Description
Enabled	
Active	Open
Problem	

3.2.2. AcDcConverter

Description	
Provides a variable DC current source to force energy directly into an EV battery stack, under tight control of the EV's battery management system.	
Typically used variables	Description
Enabled	(not commanded Out of Service)
Problem	some problem/fault exists
Tripped	A problem requiring intervention has occurred
Overload	Excessive current/power consumption
DCVoltage	measured DC voltage
DCCurrent	measured DC current
Power	measured power
Temperature	temperature of converter
FanSpeed	Speed of cooling fan(s)

3.2.3. AcPhaseSelector

Description	
Allows a specific AC phase to be selected (typically at EVSE tier) for single phase vehicle charging in order to lower overall (e.g. site) phase imbalance.	
Typically used variables	Description
Enabled	
Active	Changing
Problem	
PhaseRotation	

3.2.4. Actuator

Description	
A general purpose electro-mechanical output system, with optional completion tracking sensing. Each output should use a Variable instance key indicating the nature of the output.	
Typically used variables	Description
Enabled	

Description	
Active	Non-Default
Problem	
State	

3.2.5. AirCoolingSystem

Description	
Fans (or equivalent devices) used to provide cooling.	
Typically used variables	Description
Enabled	Cooling system enabled to run
Active	Cooling
Problem	fault: e.g. fan stalled/slow
FanSpeed	Speed of cooling fan(s)

3.2.6. AreaVentilation

Description	
Fans (or equivalent devices) used to ensure that EVs that require ventilation during charging	
Typically used variables	Description
Enabled	Area ventilation enabled
Active	Ventilating
Problem	fault: e.g. fan stalled/slow
FanSpeed	Speed of cooling fan(s)

3.2.7. BayOccupancySensor

Description	
Sensor (optical, ground loop, ultrasonic, etc.) to detect whether the associated parking/charging bay is physically vacant, or is occupied by a vehicle or other obstruction	
Typically used variables	Description
Enabled	Sensor is sensing for occupancy
Active	Occupied
Percent	percentage obstruction (for analogue sensors).

3.2.8. BeaconLighting

Description	
Beacon Lighting to help EV drivers to locate nearby charging places, and/or to determine charging availability state, usually by color variation.	
Typically used variables	Description
Enabled	Beacon Lighting operational
Enabled(Set)=0	Disable beacon lighting
Active	On
Problem	Beacon lighting fault
Percent	Lighting Level (% of maximum)
Percent(Set)=x%	Lighting Level (% of maximum)
Power	Lighting Wattage
Color	Displayed color/intensity

3.2.9. CableBreakawaySensor

Description	
A sensor that detects when a charging cable (captive or removable) has been forcibly pulled from the Charging Station.	
Typically used variables	Description
Enabled	Breakaway sensor operational
Active	Tripped
Tripped	Breakaway detected: manual check/fix required

3.2.10. CaseAccessSensor

Description	
Reports when an access door/panel is open	
Typically used variables	Description
Enabled	Access sensor is enabled to detect/report opening/closing of access door/panel
Enabled(Set)=0	Disable reporting of access
Active	Open
Tripped	An access door/panel that needs manual reset action has been activated
Problem	A fault exists in the Sensor mechanism itself

3.2.11. ChargingStation

Description	
The entire Charging Station as a logical entity	
Typically used variables	Description
Enabled	Available for use (not commanded Out of Service)
Problem	Some problem/fault exists
Identity	Charging Station identity
Tripped	A problem requiring local/manual intervention has occurred.
Overload	Excessive current/power consumption
SupplyPhases	Number of AC supply phases connected
SupplyPhases(MaxLimit)	Number of AC supply phases supported
PhaseRotation	AC wiring phase rotation
ACVoltage	Measured incoming AC voltage [per phase]
ACVoltage(MaxLimit)	Designed maximum operating AC voltage
ACCcurrent	Measured total AC current [per phase]
Power	Measured/calculated total power being consumed, including standby/ancillary loads
Power(MaxLimit)	Designed total operating load power, including standby/ancillary loads
VoltageImbalance	voltage imbalance in three phase supply
CurrentImbalance	current imbalance in three phase supply
VendorName	Charging Station vendor name (as reported in BootNotification)
Model	Charging Station model (as reported in BootNotification)
ECVariant	Engineering Change Variant
SerialNumber	Charging Station serial number
OperatingTimes	recurring operating times
ChargeProtocol	Charging Control Protocol applicable to the Charging Station
AvailabilityState	Indicates if the Charging Station is available or not (replaces the Charging Station Status values reported by the StatusNotification)
AllowNewSessionsPendingFirmwareUpdate	Indicates whether new sessions can be started on EVSEs, while Charging Station is waiting for all EVSEs to become Available in order to start a pending firmware update.

3.2.12. ChargingStatusIndicator

Description	
The Charging Status Indicator, provides visible feedback to the user about the connection and charging status of an EVSE/Connector. This is commonly in the form of multi-colored lighting.	
Typically used variables	Description
Active	Lighted
Color	Displayed color

3.2.13. ConnectedEV (updated in v1.3)

Description			
ConnectedEV is a component that represents a connected vehicle for which data is received via an ISO 15118 or CHAdeMO interface. The generic information that is received, is represented as variables of ConnectedEV. Any protocol-specific information is represented in variables of the ISO15118Ctrlr or CHAdeMOCtrlr component.			
Variable	Unit	ISO 15118-2 value	CHAdeMO value
Available	boolean	Is true when an EV is connected	
Vehicle:			
VehicleId	string	EVCCID (from SessionSetupReq) <i>Six bytes, represented as hexbinary encoded string, e.g. "010203040A0B"</i>	Vehicle ID (H'710 + H'711 + H'712) <i>Three times 8 bytes, represented as hexbinary encoded string, e.g. "010203040A0B0C0D111213141A1B1C1D212223242A2B2C2D". A concatenation of H'710 + H'711 + H'712.</i>
ProtocolAgreed	string	A string with the following comma-separated items: “<uri>,<major>,<minor>”. This is the protocol uri and version information that was agreed upon between EV and EVSE in the supportedAppProtocolReq handshake from ISO 15118. Example: "urn:iso:15118:2:2013:MsgDef,2,0"	Lowest of Chademo protocol number from EV (H'102.0) and charger (H'109.0)
ProtocolSupportedByEV	string	A string with the following comma-separated items: “<uri>,<major>,<minor>”. This is information from the supportedAppProtocolReq message from ISO 15118. Variable has multiple instances, one for each priority. Example: "urn:iso:15118:2:2013:MsgDef,2,0"	Chademo protocol number (H'102.0)
Voltage and current values:			
ACCurrent.minSet	A	EVMinCurrent	-
ACCurrent.maxSet	A	EVMaxCurrent	-
ACVoltage.maxSet	V	EVMaxVoltage	-
DCCurrent.minSet	A	-	Minimum charge current (H'100.0)
DCCurrent.maxSet	A	EVMaximumCurrentLimit	-
DCCurrent.target	A	EVTargetCurrent	Charging current request (H'102.3) <i>If HighCurrentControl is true, use the value from Charging current request (extended) (H'110.1,2).</i>
DCVoltage.minSet	V	-	Minimum battery voltage (H'100.2,3)
DCVoltage.maxSet	V	EVMaximumVoltageLimit	Maximum battery voltage (H'100.4,5)
DCVoltage.target	V	EVTargetVoltage	Target battery voltage (H'102.1,2)
Power, energy and time values:			

Variable	Unit	ISO 15118-2 value	CHAdeMO value
Power.maxSet	W	EVMaximumPowerLimit	-
EnergyCapacity	Wh	EVEnergyCapacity	Total capacity of traction battery * 100 (H'101.5,6)
EnergyImport.target	Wh	EVEnergyRequest (DC) EAmount (AC)	-
DepartureTime	dateTime	DepartureTime <i>Provided as seconds since message receipt. Convert to absolute time.</i>	-
RemainingTimeBulk	s	RemainingTimeToBulkSoC	-
RemainingTimeFull.maxSet	s	-	Maximum charging time * 60 (H'101.2)
RemainingTimeFull.actual	s	RemainingTimeToFullSoC	Estimated charging time * 60 (H'101.3)
StateOfChargeBulk	%	BulkSoC	-
StateOfCharge.maxSet	%	FullSoC	Charged rate reference constant (H'100.6)
StateOfCharge.actual	%	DC_EVStatus.EVRESSSOC	State of charge (H'102.6)
ChargingCompleteBulk	boolean	BulkChargingComplete	-
ChargingCompleteFull	boolean	ChargingComplete	-

Status values:
ChargingState

with a memberlist consisting of the following values:

* <i>BatteryOvervoltage</i>	-	Battery overvoltage (H'102.4.0)
* <i>BatteryUndervoltage</i>	-	Battery undervoltage (H'102.4.1)
* <i>ChargingCurrentDeviation</i>	FAILED_ChargingCurrentDifferential	Battery current deviation (H'102.4.2)
* <i>BatteryTemperature</i>	FAILED_RESSTemperatureInhibit	High battery temperature (H'102.4.3)
* <i>VoltageDeviation</i>	FAILED_ChargingVoltageOutOfRange	Battery voltage deviation (H'102.4.4)
* <i>ChargingSystemError</i>	FAILED_EVRESSMalfunction	Charging system error (H'102.5.2)
* <i>VehicleShiftPosition</i>	FAILED_EVShiftPosition	Vehicle shift position (H'102.5.1)
* <i>VehicleChargingEnabled</i>	-	Vehicle charging enabled (H'102.5.0)
* <i>ChargingSystemIncompatibility</i>	FAILED_ChargingSystemIncompatibility	-
* <i>ChargerConnectorLockFault</i>	FAILED_ChargerConnectorLockFault	-

3.2.14. Connector

Description	
A means to connect an EV to a Charging Station with either a socket, an attached cable & inline connector, or any wireless power transfer device.	
Typically used variables	Description
Enabled	Connector available for use (not commanded Out of Service)
Problem	problem/fault exists (e.g. over-temperature)
Tripped	A problem requiring intervention has occurred.
ConnectorType	Type of connector as defined by ConnectorEnumType in "Part 2 - Specification" plus additionally: cGBT, cChaoJi, OppCharge.
SupplyPhases	AC phases connected
SupplyPhases(MaxLimit)	AC phases Max
PhaseRotation	AC wiring phase rotation
ChargeProtocol	Charging Control Protocol applicable to the Connector
AvailabilityState	Indicates if the Connector is available or not (replaces the Status values reported by the StatusNotification)

3.2.15. ConnectorHolsterRelease

Description	
A mechanism present in a connector holster to prevent the connector from being removed inappropriately: typically unlocks connector after authorization.	
Typically used variables	Description
Enabled	
Active	Unlocked for removal/return
Problem	
State	

3.2.16. ConnectorHolsterSensor

Description	
A mechanism to report when a tethered cable connector has been removed from its normal stowage position. May be used for detection of connectors left un-holstered, and possible penalty billing.	
Typically used variables	Description
Enabled	
Active	Un-Holstered
Problem	

3.2.17. ConnectorPlugRetentionLock

Description	
Locking mechanism to retain an inserted plug, both to prevent on-load disconnection, and to prevent theft of charging cables	
Typically used variables	Description
Enabled	Retention mechanism enabled
Active	Locked
Problem	Locking Failed
Tripped	Stall protection fuse blown, etc.
Tries	(Re)tries taken on last attempt
Tries(SetLimit)	Configured auto retry count
Tries(MaxLimit)	Maximum auto retry count

3.2.18. ConnectorProtectionRelease

Description	
External protective mechanism (e.g. an external shutter or a connector holster lock mechanism) to prevent contact with conductors that may become "live" under other failure modes	
Typically used variables	Description
Enabled	Protection in effect (locked except when in use)
Active	Unlocked
Problem	Lock/Unlock mechanism fault
Tripped	protective mechanism triggered (fuse)

3.2.19. Controller

Description	
An embedded logic controller	
Typically used variables	Description
Active	Running
Problem	Controller fault
Interval[Heartbeat]	Heartbeat interval

Description	
Manufacturer	Controller manufacturer name
Model	Controller model number
ECVariant	Engineering Change variant
SerialNumber	Controller hardware serial number
VersionNumber	Hardware version number
VersionDate	Hardware version date
FirmwareVersion	Firmware version number (as reported in BootNotification)
MaxMsgElements	Array of implementation-defined limits to the number of elements of specific type that the Charging Station can accept in one message.
SelftestActive(Set)	Start self-test
SelftestActive	Self-test running

3.2.20. ControlMetering

Description	
Energy, Power, Electricity meter, used to measure energy, current, voltages etc.	
Typically used variables	Description
Power	Measured power
ACCurrent	Measured AC current [per phase]
DCCurrent	Measured DC current
DCVoltage	Measured DC voltage

3.2.21. CPPWMController

Description	
Control Pilot PWM Controller: provides and senses the IEC 61851-1 / SAE J1772 low voltage DC and PWM signalling between an EVSE and EV over a control pilot line.	
Typically used variables	Description
Enabled	
Active	Connected
Problem	CP PWM controller fault
DCVoltage	Control Pilot wire DC Voltage (0-12V)
State	IEC 61851-1 states ("A" to "E")
Percentage	1kHz Duty Cycle
SelftestActive(Set)	Start self-test
SelftestActive	Self-test running

3.2.22. DataLink

Description	
Provides a communications link from a Charging Station to a CSMS. It may use fixed infrastructure, mobile telephony data services, WiFi, or other connectivity channels.	
Typically used variables	Description
Enabled	Data link enabled
Active	Connected
Fallback	Using Backup SIM/Network Preference
Complete	Link connection terminated
Problem	Communications module or link connection fault
IMSI	International Mobile Subscriber Identity number of mobile data SIM card
ICCID	Integrated Circuit Card Identifier of mobile data SIM card.

Description	
NetworkAddress	Current network address
SignalStrength	Data signal strength/quality

3.2.23. Display

Description	
Provides information and feedback to the user.	
Typically used variables	Description
Enabled	Display configured to show information
Problem	Display fault
Color	Display color (monochrome/backlighting)
Count[HeightInChars]	Display height (characters)
Count[WidthInChars]	Display width (characters)
DataText[Visible]	Current Display Contents
State	Alphanumeric code indicating current message purpose

3.2.24. DistributionPanel

Description	
Defines the Distribution Panel, with it's fuses and connections to both Charging Stations and other Distribution Panel's.	
Common Variables	Description
InstanceName	Name of the distribution box
Fuse	Fuse (index n) is the fuse for phase Ln in Ampere.
ChargingStation	The Identity of Charging Station (index n) which is connected to this DistributionPanel. Note: this is an indexed list of Charging Station Identities, not to be confused by the Charging Station component.
ChargingStation	List of Charging Stations Identities connected to this LocalController. (not to be confused with the ChargingStation Component)
DistributionPanel	List of Distribution Panels InstanceNames connected to this LocalController. (not to be confused with the DistributionPanel Component) See the LocalController component for an example.

3.2.25. ElectricalFeed

Description	
Represents an incoming electrical connection to a Charging Station, that may be a grid/distribution network connection, or a connection to local power generation and/or storage. Each electrical feed can record the electrical and other characteristics of that feed, including power rating, fusing, upstream metering, etc. When a Charging Station has more than one electrical feed, it must represent which feed supplies each EVSE, and which feed supplies the house load of the Charging Station itself. Simple Charging Stations with only a single electrical feed may omit all electrical feed information, in which case it is inferred that all power is supplied from a single feed, and what would otherwise be ElectricalFeed data (Variables) may be reported as being associated with the ChargingStation component.	
Typically used variables	Description
Enabled	
Active	Connected
Problem	
PowerType	
Power	
Energy	
DCVoltage	
SupplyPhases	
PhaseRotation	

Description	
ACVoltage	

3.2.26. ELVSupply

Description	
Represents the low voltage power supply (typically 12V DC and often other ELV voltages) that provides operating power for controllers, relays, and other electrical components.	
Typically used variables	Description
EnergyImportRegister	Standby/house energy meter register reading
Power	instantaneous standby power consumption
Power(MaxLimit)	Design maximum standby power consumption
Fallback	Running on backup energy;
Fallback(MaxLimit): =1	has backup
StateOfCharge	backup battery SOC
Time	(estimated) operating time on backup energy

3.2.27. EmergencyStopSensor

Description	
An "Emergency Stop" button that should be pressed by the user or other nearby persons if serious faulty behavior is observed (e.g. smoke/flames from EV or Charging Station).	
Typically used variables	Description
Enabled	Emergency Stop action armed
Active	Pressed/Latched
Tripped	Needs manual reset

3.2.28. EnvironmentalLighting

Description	
Provides reporting/control of general illumination lighting in use at Charging Station.	
Typically used variables	Description
Enabled	Environmental Lighting operational
Enabled(Set)=0	Disable Environmental lighting
Active	On
Problem	Environmental lighting fault
Percent	Lighting Level (% of maximum)
Percent(Set)=x%	Lighting Level (% of maximum)
Power	Lighting Wattage
Color	Displayed color/intensity

3.2.29. EVRetentionLock

Description	
A locking mechanism on the EV side as a safety measure to prevent it being disconnected while high currents are flowing.	
Typically used variables	Description
Enabled	Retention locking detection in effect
Active	Locked to EV
Complete	Has unlocked
Problem	Lock Problem (e.g. failed to lock/unlock)

3.2.30. EVSE

Description	
The entire chain of components responsible for transporting energy from the incoming supply to the electric vehicle (or vice versa)	
Typically used variables	Description
Enabled	Ready for use (not commanded Out of Service)
Problem	some problem/fault exists
Tripped	A problem requiring intervention has occurred
Overload	Excessive current/power consumption
Evseld	The name of the EVSE in the string format as required by ISO 15118 and IEC 63119-2.
SupplyPhases	AC phases connected
PhaseRotation	AC wiring phase rotation
AllowReset	When true: EVSE can be reset individually
ACVoltage	Measured total AC voltage [per phase]
ACCCurrent	Measured total AC current [per phase]
DCVoltage	Measured total DC voltage [per phase]
DCCurrent	Measured total DC current [per phase]
Power	Measured Power
VoltageImbalance	voltage imbalance in three phase supply
CurrentImbalance	current imbalance in three phase supply
ChargeProtocol	Charging Control Protocol applicable to the EVSE
ChargingTime	Total time duration that EV is taking energy from an EVSE. Short pauses in charging (e.g. battery pre-, post-conditioning) are included
PostChargingTime	Total time since EV has taken energy from EVSE
Count[ChargingProfiles]	Charging Profiles present
Count[ChargingProfiles](MaxLimit)	Maximum Charging Profiles supported
ISO15118Evseld	The name of the EVSE in the string format as required by ISO 15118 and IEC 63119-2. Example: "DE*ICE*E*1234567890*1"

3.2.31. ExternalTemperatureSensor

Description	
Reports ambient air temperature	
Typically used variables	Description
Active	Temperature above MaxSet or MinSet
Problem	Temperature sensor fault
Temperature	Ambient temperature

3.2.32. FiscalMetering

Description	
Provides energy transfer readings that are the basis for billing.	
Typically used variables	Description
Problem	Metering Fault (e.g. read error)
EnergyImport	Energy transferred to EV during session
EnergyImportRegister	Cumulative import reading
EnergyExport	Energy transferred from EV during session
EnergyExportRegister	Cumulative export reading
Manufacturer[Meter]	Meter manufacturer name
Manufacturer[CT]	Current transformer manufacturer name
Model[Meter]	Meter model number

Description	
Model[CT]	CT model number
ECVariant	Meter engineering change variant
SerialNumber[Meter]	Meter serial number
SerialNumber[CT]	CT serial number(s)
Certificate	
OptionsSet [MeterValueAlignedData]	Set of measurands to read and report at clock-aligned time intervals while charging.
OptionsSet [TxnStoppedAlignedData]	Set of measurands to be read at clock-aligned time intervals while charging and reported in TransactionStopped

3.2.33. FloodSensor

Description	
A sensor reporting whether the Charging Station is experiencing water ingress/pooling.	
Typically used variables	Description
Enabled	Water presence/level sensing in effect
Active	Flooding
Tripped	Water level safety sensor tripped
Height	Absolute water height above reference (ground) level.
Percent	Height as percentage between reference minimum (0%) and maximum allowable (100%). Values below 0% and above 100% are possible.

3.2.34. GroundIsolationProtection

Description	
An Isolation Tester as part of their own self-test mechanisms, to confirm the isolation of floating circuitry when no Evs are connected	
Typically used variables	Description
Enabled	Electrical isolation testing enabled
Active	Leakage
Complete	Isolation test completed
Problem	Isolation fault
Impedance	Isolation resistance/impedance

3.2.35. Heater

Description	
Heater to ensure reliable operation in cold environments	
Typically used variables	Description
Enabled	Heater hardware operation enabled
Active	Heating
Problem	Heater fault
Tripped	Heater equipment permanent fault
Power	Instantaneous heater power level
Power(MaxLimit)	Maximum heater power
Power(MaxSet)	Configured heater power
Temperature(MinSet)	Cut-in temperature
Temperature(MaxSet)	Cut-out temperature

3.2.36. HumiditySensor

Description	
Reports relative air humidity	
Typically used variables	Description
Enabled	
Problem	Humidity sensor fault
Humidity	RH(%)

3.2.37. LightSensor

Description	
Reports ambient light levels.	
Typically used variables	Description
Enabled	
Problem	Lighting sensor fault
Light	The ambient light level

3.2.38. LiquidCoolingSystem

Description	
A liquid based cooling system, typically used to cool the connector cables of very high power Charging Stations.	
Typically used variables	Description
Enabled	Cooling system enabled to run
Active	Liquid circulating
Problem	
Temperature	

3.2.39. LocalAvailabilitySensor

Description	
Accepts local signal inputs controlling whether new Charging Sessions can start and/or whether ongoing sessions should continue. Typically connected to a site/building power supply, to automatically report unavailability when closed.	
Typically used variables	Description
Enabled	Local Availability input sensing in operation
Active	Out of Service
Problem	Local Availability sensing circuit error

3.2.40. LocalController

Description	
The entire Local Controller as a logical entity	
Common Variables	Description
Enabled	Available for use (not commanded Out of Service)
Problem	Some problem/fault exists
Identity	Local Controller identity
Tripped	A problem requiring local/manual intervention has occurred.
Manufacturer	Local Controller manufacturer name
Model	Local Controller manufacturer model
ECVariant	Engineering Change Variant
SerialNumber	Local Controller serial number

Description	
ChargingStation	List of Charging Stations Identities connected to this LocalController. (not to be confused with the ChargingStation Component)
DistributionPanel	<p>List of Distribution Panels InstanceNames connected to this LocalController. (not to be confused with the DistributionPanel Component)</p> <p>This can be used to describes the electrical connections in the site controlled by the Local Controller.</p> <p>An example. The incoming fuses are all 120A. Each floor has a set of 80A fuses. On the first floor, there's also a group of Charging Stations that are behind a set of 32A fuses.</p> <pre> DistributionPanel.Fuse[1] = 120 DistributionPanel.Fuse[2] = 120 DistributionPanel.Fuse[3] = 120 DistributionPanel.DistributionPanel[0] = "Level-1" DistributionPanel.DistributionPanel[1] = "Level-2" DistributionPanel["Level-1"].Fuse[1] = 80 DistributionPanel["Level-1"].Fuse[2] = 80 DistributionPanel["Level-1"].Fuse[3] = 80 DistributionPanel["Level-1"].ChargingStation[0] = "NLCP013" DistributionPanel["Level-1"].ChargingStation[1] = "NLCP014" DistributionPanel["Level-1"].ChargingStation[2] = "NLCP015" DistributionPanel["Level-1"].DistributionPanel[0] = "Level-1a" DistributionPanel["Level-1a"].Fuse[1] = 32 DistributionPanel["Level-1a"].Fuse[2] = 32 DistributionPanel["Level-1a"].Fuse[3] = 32 DistributionPanel["Level-1a"].ChargingStation[0] = "NLCP130" DistributionPanel["Level-1a"].ChargingStation[1] = "NLCP136" DistributionPanel["Level-1a"].ChargingStation[2] = "NLCP132" DistributionPanel["Level-2"].Fuse[1] = 80 DistributionPanel["Level-2"].Fuse[2] = 80 DistributionPanel["Level-2"].Fuse[3] = 80 DistributionPanel["Level-2"].ChargingStation[0] = "NLCP023" DistributionPanel["Level-2"].ChargingStation[1] = "NLCP024"</pre>

3.2.41. LocalEnergyStorage (*updated in v1.3*)

Description	
Local energy storage device	
Typically used variables	Description
EnergyCapacity	Maximum storage capacity
Identity	Local Energy Storage identity

3.2.42. OverCurrentProtection

Description	
Protects equipment by disconnecting the electrical supply when the current drawn (on any phase) exceeds the rated value to a substantial degree.	
Typically used variables	Description
Active	Tripped. Trip when over MaxSet/MaxLimit.
Operated	Breaker opened and auto-reclosed
ACCCurrent	Measured total AC current [per phase]

3.2.43. OverCurrentProtectionRecloser

Description	
Recloser mechanism of an OverCurrentProtection to perform re-arm retries after a trip, or may be set for remotely controlled re-arming on command.	
Typically used variables	Description
Enabled	Auto reclosing enabled
Active	Reclosing
Active(Set)	Initiate manual reclose
Complete	Reclose cycle completed
Problem	Recloser Fault
Mode	Reclose Mode (None, Auto, Commanded)
Tries	(Re)tries taken on last attempt
Tries(SetLimit)	Configured auto retry count
Tries(MaxLimit)	Maximum auto retry count

3.2.44. PowerContactor

Description	
Switches on and off the power to the EV after all authorization and safety requirements have been met. May have secondary contacts to report closure state.	
Typically used variables	Description
Active	Closed
Tripped	Mirror contact protection tripped
Problem	Close/Open failed

3.2.45. RCD

Description	
A Residual Current Device (US: ground fault breaker) protects human life and/or downstream equipment by quickly detecting abnormal current flows (usually indicative in earth faults) in the Charging Station, cable, or EV during charging.	
Typically used variables	Description
Tripped	Breaker opened (manual reset required)
Operated	Breaker opened and auto-reclosed

3.2.46. RCDRecloser

Description	
A motorized recloser mechanism of an RCD that may be configured to perform re-arm retries after a trip, or may be set for remotely controlled re-arming on command.	
Typically used variables	Description
Enabled	Auto reclosing enabled
Active	Reclosing in progress
Active(Set)	Initiate manual reclose
Complete	Reclose cycle completed
Problem	Recloser Fault
Tries	(Re)tries taken on last attempt
Tries(SetLimit)	Configured auto (re)try count
Tries(MaxLimit)	Maximum auto (re)try count

3.2.47. RealTimeClock

Description	
Represents realtime clock hardware that can maintain accurate date & time information in a Charging Station, even in the case of simultaneous CSMS uncontactability and power outages or resets.	
Typically used variables	
Active	RTC running OK
DCVoltage	Battery voltage
Fallback	Battery failing
Fallback(MaxLimit)	RTC has backup-power. MaxLimit = 1
Problem	RTC fault

3.2.48. ShockSensor

Description	
Measures impact forces/accelerations experienced, indicative of possible damage.	
Typically used variables	
Enabled	Shock sensing enabled
Active	Shock
Force	detected force (vector)

3.2.49. SpacesCountSignage

Description	
Electronic signage allowing a charging controller for a large charging facility to advertise counts of available spaces to passing traffic.	
Typically used variables	
Enabled	Spaces count signage enabled
Active	Not Blank
Count	

3.2.50. Switch

Description	
A general purpose electromechanical input device, with optional remote defaulting/resetting of values. Each input should use a Variable instance key indicating the nature of the input.	
Typically used variables	
Enabled	
Active	Non-Default
State	

3.2.51. TemperatureSensor

Description	
Temperature sensor at a point inside the Charging Station; multiple sensing points for a single sensing controller. Multiple sensing points for a single sensing controller may be reported using distinct Variable instance keys.	
Typically used variables	
Active	High temperature (over MaxSet)
Problem	Internal temperature sensor fault
Temperature	Enclosure temperature

3.2.52. TiltSensor

Description	
Measures Tilt angle from normal reference position (normally 90 degree vertical).	
Typically used variables	Description
Enabled	Tilt sensing enabled
Active	Tilted
Angle	Measured tilt (vector) from vertical

3.2.53. TokenReader

Description	
An authorization token reader (e.g. RFID)	
Typically used variables	Description
Enabled	Token reader enabled
Enabled(Set)=0	Token reader disabled: allow charging without token authentication/authorization
Operated	token data read event
Problem	token reader fault
Token	String read by TokenReader
TokenType	Type of token as IdTokenEnumType

3.2.54. UpstreamProtectionTrigger

Description	
Circuitry designed to trigger the disconnection of power to the structure by an upstream protection device after a severe problem has been detected	
Typically used variables	Description
Enabled	Upstream protection enabled
Active(Set)	Force triggering of upstream protection
Tripped	Upstream protection triggered
Problem	Upstream protection fault

3.2.55. UIInput

Description	
A logical input mechanism (e.g. set of buttons) that is part of a UI whose use may be communicated to the CSMS (in near real time). May support momentary inputs ("Operated") or modal state ("Active"). Multiple input sources should use explicit Variable instance keys (where the input function is key name).	
Typically used variables	Description
Enabled	UI input enabled
Operated	
Active	

3.2.56. VehicleIdSensor

Description	
Reports an identifier associated with a vehicle occupying a charging bay. The identifier may be a vehicle registration number via ANPR hardware, a VIN, or other local identifier of the vehicle based on medium range/active RFID, or any other relevant technology and result.	
Typically used variables	Description
Enabled	VehicleIdSensor enabled
Active	Processing

3.3. Summary List of Standardized Components

Following is a list that sums up all above-mentioned standardized component names.

Component	Description
AlignedDataCtrlr	Logical Component responsible for configuration relating to the reporting of clock-aligned meter data.
AuthCtrlr	Logical Component responsible for configuration relating to the use of authorization for Charging Station use.
AuthCacheCtrlr	Logical Component responsible for configuration relating to the use of a local cache for authorization for Charging Station use.
CHAdeMOCtrlr	A CHAdeMO Controller component communicates with an EV using the wired CANbus protocol to exchange information and control charging using the CHAdeMO protocol
ClockCtrlr	Provides a means to configure management of time tracking by Charging Station.
DeviceDataCtrlr	Logical Component responsible for configuration relating to the exchange and storage of Charging Station Device Model data.
DisplayMessageCtrlr	Logical Component responsible for configuration relating to the display of messages to Charging Station users.
ISO15118Ctrlr	Communicates with an EV to exchange information and control charging using the ISO 15118 protocol.
LocalAuthListCtrlr	Logical Component responsible for configuration relating to the use of Local Authorization Lists for Charging Station use.
MonitoringCtrlr	Logical Component responsible for configuration relating to the exchange of monitoring event data.
OCPPCommCtrlr	Logical Component responsible for configuration relating to information exchange between Charging Station and CSMS.
ReservationCtrlr	Logical Component responsible for configuration relating to reservations.
SampledDataCtrlr	Logical Component responsible for configuration relating to the reporting of sampled meter data.
SecurityCtrlr	Logical Component responsible for configuration relating to security of communications between Charging Station and CSMS.
SmartChargingCtrlr	Logical Component responsible for configuration relating to smart charging.
TariffCostCtrlr	Logical Component responsible for configuration relating to tariff and cost display.
TxCtrlr	Logical Component responsible for configuration relating to transaction characteristics and behaviour.
AccessBarrier	Allows physical access of vehicles to a charging site to be controlled.
AcDcConverter	Provides a variable DC current source to force energy directly into an EV battery stack, under tight control of the EV's battery management system.
AcPhaseSelector	Allows a specific AC phase to be selected (typically at EVSE tier) for single phase vehicle charging in order to lower overall (e.g. site) phase imbalance.
Actuator	A general purpose electro-mechanical output system, with optional completion tracking sensing. Each output should use a Variable instance key indicating the nature of the output.
AirCoolingSystem	Fans (or equivalent devices) used to provide cooling.
AreaVentilation	Fans (or equivalent devices) used to ensure that EVs that require ventilation during charging
BayOccupancySensor	Sensor (optical, ground loop, ultrasonic, etc.) to detect whether the associated parking/charging bay is physically vacant, or is occupied by a vehicle or other obstruction
BeaconLighting	Beacon Lighting to help EV drivers to locate nearby charging places, and/or to determine charging availability state, usually by color variation.
CableBreakawaySensor	A sensor that detects when a charging cable (captive or removable) has been forcibly pulled from the Charging Station.
CaseAccessSensor	Reports when an access door/panel is open
ChargingStation	The entire Charging Station as a logical entity
ChargingStatusIndicator	The Charging Status Indicator, provides visible feedback to the user about the connection and charging status of an EVSE/Connector. This is commonly in the form of multi-colored lighting.
ConnectedEV	ConnectedEV is a component that represents a connected vehicle for which data is received via an ISO 15118 or CHAdeMO interface. The generic information that is received, is represented as variables of ConnectedEV. Any protocol-specific information is represented in variables of the ISO15118Ctrlr or CHAdeMOCtrlr component.
Connector	A means to connect an EV to a Charging Station with either a socket, an attached cable & inline connector, or any wireless power transfer device.

Appendix 3. Standardized Components

Component	Description
ConnectorHolsterRelease	A mechanism present in a connector holster to prevent the connector from being removed inappropriately: typically unlocks connector after authorization.
ConnectorHolsterSensor	A mechanism to report when a tethered cable connector has been removed from its normal stowage position. May be used for detection of connectors left un-holstered, and possible penalty billing.
ConnectorPlugRetentionLock	Locking mechanism to retain an inserted plug, both to prevent on-load disconnection, and to prevent theft of charging cables
ConnectorProtectionRelease	External protective mechanism (e.g. an external shutter or a connector holster lock mechanism) to prevent contact with conductors that may become 'live' under other failure modes
Controller	An embedded logic controller
ControlMetering	Energy, Power, Electricity meter, used to measure energy, current, voltages etc.
CPPWMController	Control Pilot PWM Controller: provides and senses the IEC 61851-1 / SAE J1772 low voltage DC and PWM signalling between an EVSE and EV over a control pilot line.
DataLink	Provides a communications link from a Charging Station to a CSMS. It may use fixed infrastructure, mobile telephony data services, WiFi, or other connectivity channels.
Display	Provides information and feedback to the user.
DistributionPanel	Defines the Distribution Panel, with it's fuses and connections to both Charging Stations and other Distribution Panel's.
ElectricalFeed	Represents an incoming electrical connection to a Charging Station, that may be a grid/distribution network connection, of a connection to local power generation and/or storage. Each electrical feed can record the electrical and other characteristics of that feed, including power rating, fusing, upstream metering, etc. When a Charging Station has more than one electrical feed, it must represent which feed supplies each EVSE, and which feed supplies the house load of the Charging Station itself. Simple Charging Stations with only a single electrical feed may omit all electrical feed information, in which case it is inferred that all power is supplied from a single feed, and what would otherwise be ElectricalFeed data (Variables) may be reported as being associated with the ChargingStation component.
ELVSupply	Represents the low voltage power supply (typically 12V DC and often other ELV voltages) that provides operating power for controllers, relays, and other electrical components.
EmergencyStopSensor	An 'Emergency Stop' button that should be pressed by the user or other nearby persons if serious faulty behavior is observed (e.g. smoke/flames from EV or Charging Station).
EnvironmentalLighting	Provides reporting/control of general illumination lighting in use at Charging Station.
EVRetentionLock	A locking mechanism on the EV side as a safety measure to prevent it being disconnected while high currents are flowing.
EVSE	The entire chain of components responsible for transporting energy from the incoming supply to the electric vehicle (or vice versa)
ExternalTemperatureSensor	Reports ambient air temperature
FiscalMetering	Provides energy transfer readings that are the basis for billing.
FloodSensor	A sensor reporting whether the Charging Station is experiencing water ingress/pooling.
GroundIsolationProtection	An Isolation Tester as part of their own self-test mechanisms, to confirm the isolation of floating circuitry when no EVs are connected
Heater	Heater to ensure reliable operation in cold environments
HumiditySensor	Reports relative air humidity
LightSensor	Reports ambient light levels.
LiquidCoolingSystem	A liquid based cooling system, typically used to cool the connector cables of very high power Charging Stations.
LocalAvailabilitySensor	Accepts local signal inputs controlling whether new Charging Sessions can start and/or whether ongoing sessions should continue. Typically connected to a site/building power supply, to automatically report unavailability when closed.
LocalController	The entire Local Controller as a logical entity
LocalEnergyStorage	Energy storage
OverCurrentProtection	Protects equipment by disconnecting the electrical supply when the current drawn (on any phase) exceeds the rated value to a substantial degree.
OverCurrentProtectionRecloser	Recloser mechanism of an OverCurrentProtection to perform re-arm retries after a trip, or may be set for remotely controlled re-arming on command.

Appendix 3. Standardized Components

Component	Description
PowerContactor	Switches on and off the power to the EV after all authorization and safety requirements have been met. May have secondary contacts to report closure state.
RCD	A Residual Current Device (US: ground fault breaker) protects human life and/or downstream equipment by quickly detecting abnormal current flows (usually indicative in earth faults) in the Charging Station, cable, or EV during charging.
RCDRecloser	A motorized recloser mechanism of an RCD that may be configured to perform re-arm retries after a trip, or may be set for remotely controlled re-arming on command.
RealTimeClock	Represents realtime clock hardware that can maintain accurate date & time information in a Charging Station, even in the case of simultaneous CSMS uncontactability and power outages or resets.
ShockSensor	Measures impact forces/accelerations experienced, indicative of possible damage.
SpacesCountSignage	Electronic signage allowing a charging controller for a large charging facility to advertise counts of available spaces to passing traffic.
Switch	A general purpose electromechanical input device, with optional remote defaulting/resetting of values. Each input should use a Variable instance key indicating the nature of the input.
TemperatureSensor	Temperature sensor at a point inside the Charging Station, multiple sensing points for a single sensing controller. Multiple sensing points for a single sensing controller may be reported using distinct Variable instance keys.
TiltSensor	Measures Tilt angle from normal reference position (normally 90 degree vertical).
TokenReader	An authorization token reader (e.g. RFID)
UpstreamProtectionTrigger	Circuitry designed to trigger the disconnection of power to the structure by an upstream protection device after a severe problem has been detected
UIInput	A logical input mechanism (e.g. set of buttons) that is part of a UI whose use may be communicated to the CSMS (in near real time). May support momentary inputs ('Operated') or modal state ('Active'). Multiple input sources should use explicit Variable instance keys (where the input function is key name).
VehicleIdSensor	Reports an identifier associated with a vehicle occupying a charging bay. The identifier may be a vehicle registration number via ANPR hardware, a VIN, or other local identifier of the vehicle based on medium range/active RFID, or any other relevant technology and result.

Appendix 4. Standardized Variables

This is a non-exhaustive list of Standardized Variables that SHALL be used when the Charging Station and CSMS want to exchange information about a Variable. See also Part 1, paragraph 4.5.

Variables that are specific to a Controller Component are not included in this list, but are part of section 3.1 Controller Components.

Name	DataType	Unit	Description
ACCCurrent	decimal	A	RMS AC Current (in amperes). For 3-phase circuits, each phase (and optional neutral) is represented by a Variable instance equal to a value of the PhaseEnumType (e.g. L1,N). Unkeyed values reported for a Component declared to be multi-phase are assumed to be an average of all per-phase readings and written values are common per-phase settings. Example(s): ChargingStation: Total AC current consumption (all EVSE's, ancillaries), EVSE: Total current consumed by EVSE: includes losses (AC->DC) and EVSE specific ancillaries (e.g. fans), ElectricalFeed: Inflow AC current on feed
Active	boolean		Component is in its non-resting / active state: e.g: On, Engaged, Locked. Some Components may have secondary functions that have corresponding Active Variables with an explicit Variable instance., Note: Monitoring of changes in the Active state of any Component can be specified by setting Delta monitoring on the boolean value with a delta values of 1. Setting/clearing an Active Variable activates/stops the associated functionality, where remotely controllable. Only components that are Available and Enabled can be in the Active state.
ACVoltage	decimal	V	RMS AC Voltage (in volts). For 3-phase circuits, each phase (and optional neutral) is represented by a Variable instance equal to a value of the PhaseEnumType (e.g. L1,N). Unkeyed values reported for a Component declared to be multi-phase are assumed to be an average of all per-phase readings and written values are common per-phase settings. Example(s): ElectricalFeed: Input Voltage
AllowReset	boolean		Component can be reset. Can be used to announce that an EVSE can be reset individually.
Angle	decimal	Deg	Angle(s) relative to normal/design idle position. Multiple Variable instance values may be used to indicate angular position in multiple axes (e.g. Left-Right, Forward-Back).
Attempts	integer		Number of attempts (INCLUDING the original attempt) in the last successful or attempted, cycle of operation. Applies typically to self-monitoring motorized electro-mechanical equipment, etc. {Null}: Unknown, 0: Not Attempted/Not allowed, 1: Single attempt/No retries [allowed], 2-N: [up to] N tries [allowed]
AvailabilityState	OptionList		A value of ConnectorStatusEnumType (See part 2): replicates ConnectorStatus values reported in StatusNotification messages.
Available	boolean		The Component exists and is locally configured/wired for use, but might not be (remotely) Enabled.
Certificate	string		Digital Certificate (in Base64 encoding)
ChargeProtocol			The Charging Control Protocol applicable to a Connector. CHAdeMO: CHAdeMO protocol, ISO15118: ISO15118 V2G protocol (wired or wireless) as used with CCS, CPPWM: IEC61851-1 / SAE J1772 protocol (ELV DC & PWM signalling via Control Pilot wire), Uncontrolled: No charging power management applies (e.g. Schuko socket), Undetermined: Yet to be determined (e.g. before plugged in), Unknown: Not determinable, NOTE: ChargeProtocol is distinct from and orthogonal to connectorType.
ChargingCompleteBulk	boolean		Charging up to StateOfChargeBulk has completed.
ChargingCompleteFull	boolean		Charging up to StateOfCharge.maxSet has completed.
ChargingTime	decimal	s	Time from earliest to latest substantive energy transfer
Color	string		Standard 24 bit hexadecimal RGB values. Reg Green Blue color intensity, expressed as standard 24 bit hexadecimal RGB values: 3 00-FF (0-255), in order RRGGBB). E.g. 000000: Black, FF0000: Red, 00FF00: Green, 0000FF: Blue, FFFF00:Yellow, FFFFFF: White, 008000: Medium intensity green.

Appendix 4. Standardized Variables

Name	DataType	Unit	Description
Complete	boolean		Component's operation cycle has completed. Used only in event notifications, where it is always true.
ConnectedTime	decimal	s	Time since logical connection established
ConnectorType	OptionList		A value of ConnectorEnumType (See part 2) plus additionally: cGBT, cChaoJi, OppCharge. Specific type of connector, including sub-variant information. Note: Distinct and orthogonal to Charging Protocol, Power Type, Phases.
Count	integer		General purpose integer count variable for Component state reporting
Currency	string		Currency in a ISO 4217 formatted currency code.
CurrentImbalance	decimal	Percent	Percentage current imbalance in an AC three phase supply.
DataText	string		Text associated with a Component, e.g. a Display.
DateTime	dateTime		Point in time value, in [RFC3339] datetime format. Time zone optional.
DCCurrent	decimal	A	DC Current (in amperes). May be an instantaneous measurement, or a period average, depending on context/equipment.
DCVoltage	decimal	V	DC Voltage (volts). May be an instantaneous measurement, or a period average, depending on context/equipment.
DepartureTime	dateTime		Time in [RFC3339] datetime format, when an EV intends to leave the charging station.
ECVariant	string		Production series variants reflecting internal design changes or sub-component substitutions not affecting external functionality.
Enabled	boolean		The Component is Enabled for operation. For Available components that cannot be selectively (remotely) enabled / disabled, this value is always true. Note: Available cannot be false if Enabled is true, so during inventory reporting, Enabled=1 also logically states Available=true
Energy	decimal	Wh	Energy quantity (in Wh) for reporting/configuring values related to stored energy (i.e. not transferred energy).
EnergyCapacity	decimal	Wh	Energy capacity in Wh of an energy storage device.
EnergyExport	decimal	Wh	Total energy transferred: e.g. from EV during (ongoing or terminated) charging session (in wH by default)
EnergyExportRegister	decimal	Wh	Cumulative export kWh register value, such as from a (certified) fiscal energy meter.
EnergyImport	decimal	Wh	Total energy transferred.
EnergyImportRegister	decimal	Wh	Cumulative export kWh register value, such as from a (certified) fiscal energy meter.
Entries	integer		General purpose variable for reporting/managing numbers of entries in repetitive data structures. maxLimit characteristic reports maximum possible entries.
Evseld	string		EVSE ID in string format as used in ISO 15118 and IEC 63119-2
Fallback	boolean		Component is operating in a fallback, or backup mode. In inventory reports, a Value of 1 for the maxLimit characteristic indicates that the component can enter a fallback state (i.e. a fallback mode is present).
FanSpeed	decimal	RPM	Fan Speed (in RPM). A value of 0 represents stopped/stalled. An empty value indicates that fan speed cannot be read.
FirmwareVersion	string		Version number of firmware.
Force	decimal	N	Reports (impact) force/ acceleration values (estimates) in one or more directions, in units of Newtons or "g". Multiple force readings in different (orthogonal) dimensions may be reported using Variable instance values, such as Down, Right, Forward.
Formats	MemberList		List of message formats supported by this Charging Station. Possible values: ASCII, HTML, URI, UTF-8.
Frequency	decimal	Hz	Frequency of AC power, signal, or component operation.
FuseRating	decimal	A	Current rating of a fuse/breaker. Variable instances keyed by phase identifier (L1/L2/L3/N).
Height	decimal	m	Height above(+)below(-) reference level (ground level unless context demands otherwise).
Humidity	decimal	RH	The relative humidity in %.

Appendix 4. Standardized Variables

Name	DataType	Unit	Description
Hysteresis	decimal	Percent	Specifies the width of a 'dead band' (as a percentage of the threshold) around the central value of a threshold setting (e.g. MinSet, MaxSet, monitor thresholds) to avoid repeated triggering when the measured quantity lies close to the threshold and is subject to small variations.
ICCID	string		ICCID (Integrated Circuit Card IDentifier) of mobile data SIM card.
Impedance	decimal	Ohm	Impedance: Primary value is real (resistive only) impedance. Where a complex impedance is to be reported, the imaginary part (reactance) must be represented with a separate Variable instance value of 'reactance'. Reactance values are expressed at the (nominal) relevant operating frequency of the Component (e.g. 50/60Hz for mains electricity feed).
IMSI	string		IMSI (International Mobile Subscriber Identity) number of mobile data SIM card
Interval	integer	s	Minimum Interval (in seconds) between (attempted) operations.
Length	decimal	m	General Purpose linear distance measure.
Light	decimal	lx	(Ambient) light level. The value is in Lux.
Manufacturer	string		Component Manufacturer name
Message	string		Specific stored message for display.
MinimumStatusDuration	integer	s	Minimum duration that a Charging Station or EVSE status is stable before StatusNotificationRequest is sent to the CSMS.
Mode	string		Operating mode string from among valid options (communicated by OptionList, etc. during capability/configuration discovery).
Model	string		Manufacturer's Model code/number of Component, including suffixes etc. to identify functional, regional or linguistic variation, but NOT engineering change level internal variation not affecting external behaviour, etc.
NetworkAddress	string		Current network address of a Component.
Operated	boolean		The Component operated in an instantaneous, transient, or immediately self-resetting pattern. Used only in event notifications, where it is always true.
OperatingTimes	string		Recurring operating times in iCalendar RRULE format.
Overload	boolean		Component is in Overload state.
Percent	decimal	Percent	Generic dimensionless value reporting/setting value.
PhaseRotation	OptionList		The phase wiring of Component, relative to its upstream feed Component/device. This variable describes the phase rotation of a Component relative to its parent Component, using a three letter string consisting of the letters: R, S, T and x. The letter 'R' can be identified as phase 1 (L1), 'S' as phase 2 (L2), 'T' as phase 3 (L3). The lower case 'x' is used to designate a phase that is not connected. An empty string means that phase rotation is not applicable or not known.
PostChargingTime	decimal	s	Elapsed time in seconds since last substantive energy transfer
Power	decimal	W,kW	Instantaneous (real) Power (measured/calculated, including power factor for AC). Where a component (e.g. AC to DC Power Converter) has multiple power measurements, the default (unkeyed) instance is "input" power.
Problem	boolean		Component itself has a 'Problem' condition that impacts in any significant way on its normal operation. By definition, 'Problem' state includes (logical OR) 'Fault' state. 'Problem' specifically INCLUDES inability to operate that is propagated (up/down/sideways) from any other associated/connected/containing/contained Component.
Protecting	boolean		Applies to 'sensor' type Components that have an associated protection capability, whereby they can disconnect power (e.g. using the main PowerContactor) if the sensed quantity is outside preset/configured limits. If Protecting is true, the Component is actively preventing/interrupting charging.
RemainingTimeBulk	integer	s	Number of seconds remaining to charge to bulk state of charge, given by StateOfChargeBulk.
RemainingTimeFull	integer	s	Number of seconds remaining to charge to 100% state of charge.
SeccId	string		The name of the SECC in the string format as required by ISO 15118.

Appendix 4. Standardized Variables

Name	DataType	Unit	Description
SerialNumber	string		Serial number of Component.
SignalStrength	decimal	dBm	(Radio/Wired/Optical) data signal strength, in ASU (typically 0-31 or 99 for unknown). Or dbmW (typically -140 to -50).
State	string		A state code or name identifier string, to allow the internal state of components to be reported and/or controlled
StateOfCharge	decimal	Percent	Energy Storage Device (e.g. battery) state of charge, expressed as a percentage of nominal design 0-100% operating range. The value of StateOfCharge.maxSet represents the maximum state of charge for a full battery and is usually at or near 100%.
StateOfChargeBulk	decimal	Percent	Energy Storage Device (e.g. battery) state of charge up to which fast charging is possible. Above this percentage charging speed will drop significantly.
Storage	integer	B	In bytes. Amount of storage occupied. Storage(maxLimit) specifies absolute limit Storage(MaxSet) restricts usage to specified Max, if supported.
SupplyPhases	integer		Number of alternating current phases connected/available. 1 or 3 for AC, 0 means DC (no alternating phases). Null value indicates that the number of phases (e.g. in use) is unknown.
Suspending	boolean		If Suspending is true, the Component can be currently suspending charging.
Suspension	boolean		Applies to 'sensor' type Components that have a charging suspension capability, typically for safety or equipment protection reasons. If Suspension is true, the component can suspend charging when the sensed quantity is outside preset/configured limits.
Temperature	decimal	Celsius, Fahrenheit	Temperature(s) of component (in Celsius, by default). Components may have multiple indexed temperature sensors.
Time	dateTime		Point in time value, in ISO 8601 datetime format. Time zone optional.
TimeOffset	string		A Time Offset with respect to Coordinated Universal Time (aka UTC or Greenwich Mean Time) in the form of an [RFC3339] time (zone) offset suffix, including the mandatory "+" or "-" prefix.
Timeout	decimal	s	Generic timeout value for Component operation (in seconds).
Token	string		String of bytes representing an ID token.
TokenType	OptionList		Type of Token. Value is one of IdTokenEnumType.
Tries	integer		Number of attempts done by a Component.
Tripped	boolean		Single-shot device requires explicit intervention to re-prime/activate to normal.
VehicleId	string		ID that EV provides to charging station. Encoded as a hexbinary string. In ISO 15118 the EVCCID is 6 bytes (MAC address), in CHAdeMO the vehicle id can be 24 bytes.
VersionDate	dateTime		[RFC3339]
VersionNumber	string		Version number of hardware
VoltageImbalance	decimal	Percent	Percentage voltage imbalance in three phase supply.

Appendix 5. Reason Codes

The table below provides a list of standardized reason codes that can be used in the optional StatusInfo element of a response.

For each reason code, some messages that might typically return them are shown. This is not an exhaustive list and only indicative.

StatusInfo is optional. Any implementation should be able to function properly without StatusInfo, because every message has the response code values that are essential to perform the function. The *reasonCode* and *additionalInfo* in StatusInfo are meant to provide more insight on what is happening and maybe allow for some automatic recovery.

IMPORTANT

The existence of a reason code in this table does not imply a requirement to use it nor does it imply a requirement to any of the mentioned messages.

Reason Code	Description	Typically used for
CSNotAccepted	BootNotification of Charging Station has not (yet) been accepted by CSMS.	RequestStartTransaction, RequestStopTransaction
DuplicateProfile	A charging profile with same <i>stackLevel - chargingProfilePurpose</i> combination already exists on the Charging Station and has an overlapping validity period.	SetChargingProfile
DuplicateRequestId	A <i>requestId</i> is provided, that has already been used for this type of request.	UpdateFirmware, PublishFirmware and requests for reports.
FixedCable	The connector has its own fixed cable that cannot be unlocked.	UnlockConnector
FwUpdateInProgress	Operation is not possible, because a firmware update is in progress.	Reset
InternalError	Operation cannot be completed due to an internal error.	(generic)
InvalidCertificate	Provided certificate is invalid.	CertificateSigned, InstallCertificate
InvalidCSR	Provided CSR is invalid	SignCertificate
InvalidIdToken	Provided <i>idToken</i> is not valid.	RequestStartTransaction
InvalidMessageSequence	Message should not be sent at this moment in current scenario.	(generic), SetChargingProfile with ISO15118
InvalidProfile	Provided <i>chargingProfile</i> contains invalid elements.	SetChargingProfile, RequestStartTransaction
InvalidSchedule	Provided <i>chargingSchedule</i> contains invalid elements.	SetChargingProfile, RequestStartTransaction
InvalidStackLevel	Provided value for <i>stackLevel</i> is invalid.	SetChargingProfile
InvalidURL	Provided URL is invalid.	UpdateFirmware, PublishFirmware
InvalidValue	An invalid value has been provided.	(generic)
MissingDeviceModelInfo	Information needed for operation is missing from Device Model	(generic)
MissingParam	A parameter that is required for the request is missing.	(generic)
NoCable	No cable is connected at this time.	UnlockConnector
NoError	No error has occurred, but some extra information is in <i>additionalInfo</i> .	(generic)
NotEnabled	Feature is not enabled.	ClearCache
NotFound	No object(s) found that match a provided ID or criteria.	ClearVariableMonitoring, CustomerInformation, GetChargingProfiles, GetDisplayMessages, GetInstalledCertificateIds, GetReport
OutOfMemory	Operation not possible, because system does not have enough memory.	(generic)
OutOfStorage	Operation not possible, because system does not have enough storage.	(generic)
ReadOnly	Targeted variable is read-only and cannot be set.	SetVariables

Reason Code	Description	Typically used for
TooLargeElement	Provided element is too large to handle.	CertificateSigned, InstallCertificate
TooManyElements	Too many elements have been provided.	SetChargingProfile, SetVariables, SendLocalList
TxInProgress	A transaction is in progress.	ChangeAvailability, Reset, RequestStartTransaction
TxNotFound	There is no such transaction.	RequestStopTransaction, SetChargingProfile
TxStarted	A transaction had already started (e.g. due to cable being plugged in).	RequestStartTransaction
UnknownConnectorId	Connector Id is not known on EVSE	ChangeAvailability, UnlockConnector
UnknownConnectorType	Connector type is not known on EVSE	ReserveNow
UnknownEvse	EVSE is not known on Charging Stations	ChangeAvailability, ReserveNow, RequestStartTransaction
UnknownTxId	Provided <i>transactionId</i> is not known.	RequestStopTransaction
Unspecified	No reason is specified, but some extra information is in <i>additionalInfo</i>	(generic)
UnsupportedParam	A parameter was provided that is not supported.	(generic)
UnsupportedRateUnit	A <i>chargingRateUnit</i> is provided that is not supported.	SetChargingProfile
UnsupportedRequest	This request is not supported.	(generic)
ValueOutOfRange	Provided value is out of range.	SetVariables, SetVariableMonitoring
ValuePositiveOnly	Provided value is not greater than zero.	(generic)
ValueTooHigh	Provided value is too high.	(generic)
ValueTooLow	Provided value is too low.	(generic)
ValueZeroNotAllowed	Provided value cannot be zero.	(generic)
WriteOnly	Targeted variable is write-only and cannot be read.	GetVariables



OCPP 2.0.1

Part 2 Edition 2 - Errata

v1.0, 2023-06-30

Table of Contents

1. Scope	2
1.1. Terminology and Conventions	2
2. General	3
3. Use case A Security	3
3.1. Page 23 - (v1) Requirement A00.FR.316: Make clear that InvalidTLSVersion must be queued [689]	3
4. Use case B Provisioning	4
4.1. Page 61 - (v1) Requirement B08.FR.19 and N02.FR.15 are ambiguous w.r.t. evse and instance wildcards [676]	4
4.2. Page 62 - (v1) Use case B09/B10: Extended scenario description [683]	5
5. Use case C Authorization	6
5.1. Page 90 - (v1) C07 requirements for certificateStatus missing [680]	7
5.2. Page 97 - (v1) Requirement C10.FR.06 needs to be removed [685]	8
5.3. Page 102 - (v1) Requirement C13.FR.04 enhanced [701]	8
6. Use Case E Transactions	9
6.1. Page 147 - (v1) Use case E07 - Scenario description step order incorrect [704]	9
6.2. Page 148 - (v1) Use case E07: Wrong triggerReason shown in sequence diagram fig. 56 [687]	9
6.3. Page 150 - (v1) Use case E07: Clarify 'normal' and 'correct' for stoppedReason [693]	10
7. Use Case F Remote Control	11
7.1. Page 180 - (v1) Requirement F03.FR.03 contains wrong precondition [700]	11
7.2. Page 187 - (v1) Requirement F06.FR.12 is too strict [707]	12
8. Use Case G Availability	12
8.1. Page 192 - (v1) G01.FR.08 contradicts H01.FR.24 [692]	12
9. Use Case H Reservation	13
9.1. Page 205 - (v1) Missing option to send NotifyEvent instead of StatusNotification [699]	13
9.2. Page 209 - (v1) Remark about authorization in use case H03 [711]	14
9.3. Page 210 - (v1) Requirement H03.FR.08 is not clear about groupIdToken lookup [684]	14
10. Use Case J Meter Values	15
10.1. Page 228 - (v1) Requirement J01.FR.14 is unclear that meter values for all EVSEs must be sent [674]	15
10.2. Page 230 - (v1) Requirement J02.FR.10 refers to all TransactionEventRequest messages, but should be specific to only eventType = Updated [705]	15
10.3. Page 231 - (v1) J01 misses requirement that meter value must be for current transaction [673]	16
11. Use Case K Smart Charging	16
11.1. Page 238 - (v1) Text in section 3.3 does not match ChargingProfileKindEnumType description [708]	16
12. Use Case L FirmwareManagement	16
12.1. Page 287 - (v1) Improved title of figure 119 [695]	16
13. Use Case M ISO 15118 CertificateManagement	16
13.1. Page 310 - (v1) M04.FR.07 has an incorrect requirement definition [703]	16
14. Use Case N Diagnostics	17
14.1. Page 317 - (v1) N01.FR.10 not clear when to report UploadFailure [696]	17
14.2. Page 331 - (v1) Requirement N09.FR.04 has been rephrased [688]	17
15. Messages	18
15.1. Page 353 - (v1) Clarification for use of certificate and iso15118CertificateHashData in AuthorizeRequest [675]	18
15.2. Page 381 - (v1) Updated description for idToken in TransactionEventRequest [709]	18
16. Data Types	19
16.1. Page 386 - (v1) issuerKeyHash in CertificateHashDataType must be type identifierString [691]	19
16.2. Page 396 - (v1) NetworkConnectionProfileType [683]	19
17. Enumerations	19
17.1. Page 419 - (v1) Description for idTokenEnumType MacAddress [664]	20
18. Referenced Components and Variables	20
18.1. Page 436 - (v1) Websocket-related variables in Part 4 [690]	20
18.2. Page 444 - (v1) SecurityCtrlr.BasicAuthPassword and Identity should have dataType=string	20
18.3. Page 452 - (v1) Incomplete description TxStopPoint Authorized and PowerPathClosed [704]	21
19. Appendix 1	21
19.1. Page 2 - (v1) InvalidFirmwareSignature/SigningCertificate are critical security events [682]	21
20. Appendix 3	22
20.1. Page 9 - (v1) OCPPCommCtrlr.ActiveNetworkProfile must be of type integer [697]	22
20.2. Page 10 - (v1) SecurityCtrlr.BasicAuthPassword and Identity should have dataType=string [698]	22

Version History

Version	Date	Description
1.0	2023-06-30	Errata OCPP 2.0.1 edition 2

1. Scope

This document contains errata on "part 2: Specification" and "part 2: Appendices" of the OCPP 2.0.1 Edition 2 documentation. These errata have to be read as an addition to the release of OCPP 2.0.1 Edition 2.

The errata do not affect any schemas of OCPP messages. Certain errata do contain changes to requirements or even new requirements, but only in cases where a requirement contains an obvious error and would not or could not be implemented literally. New requirements are only added when they were already implicitly there. These changes have been discussed in or were proposed by the Technology Working Group of the Open Charge Alliance.

The appendices of the OCPP specification can be updated without requiring a new OCPP release. This mainly concerns the components and variables of the OCPP device model, which can be extended with new components or variables, as long as they are optional.

1.1. Terminology and Conventions

Bold: when needed to clarify differences, bold text might be used.

The errata entries are sorted by page number of the affected section of the specification document. When an errata entry affects multiple parts of the specification, then the various changes are grouped together with subsections referring to the pages affected by those changes.

This is version 1 of the errata. The errata of this version are marked with "(v1)" in the section title. Whenever a second version of the errata is released, then its section titles will be marked with "(v2)".

Where possible the issue number by which it was reported, is added in square brackets at the end of the section title, e.g. "[349]". For retrieval of the issue in the issue tracking system prefix the number with "OCPP20M", like "[OCPP20M-349]".

2. General

3. Use case A Security

3.1. Page 23 - (v1) Requirement A00.FR.316: Make clear that InvalidTLSVersion must be queued [689]

Requirement A00.FR.316 states that a security event InvalidTLSVersion is triggered and connection must be closed. It is not clear from this requirement that this must also be sent as a security event notification when a connection to CSMS is made. This is stated in use case A04. Obviously, once CSMS accepts the connection, the InvalidTLSVersion condition no longer applies at this time, but the event must still be reported.

Changed requirement

	ID	Precondition	Requirement definition
Old text	A00.FR.316	A00.FR.314 AND The Charging Station detects that the CSMS only allows connections using an older version of TLS, or only allows SSL	The Charging Station SHALL trigger an InvalidTLSVersion security event AND terminate the connection (See part 2 appendices for the full list of security events).
New text	A00.FR.316	A00.FR.314 AND The Charging Station detects that the CSMS only allows connections using an older version of TLS, or only allows SSL	<p>The Charging Station SHALL trigger an InvalidTLSVersion security event AND terminate the connection (See part 2 appendices for the full list of security events).</p> <p>NOTE: This is a critical security event that will need to be queued and sent to CSMS once a successful connection has been made, as described in use case A04.</p> <p>A security event only needs to be sent once for repeated failed connection attempts, in order to avoid overflow to the offline queue.</p>

3.1.1. Page 25 - Requirement A00.FR.419

Changed requirement

	ID	Precondition	Requirement definition
Old text	A00.FR.419	A00.FR.417 AND The Charging Station detects that the CSMS only allows connections using an older version of TLS, or only allows SSL	The Charging Station SHALL trigger an InvalidTLSVersion security event AND terminate the connection (See part 2 appendices for the full list of security events).
New text	A00.FR.419	A00.FR.417 AND The Charging Station detects that the CSMS only allows connections using an older version of TLS, or only allows SSL	<p>The Charging Station SHALL trigger an InvalidTLSVersion security event AND terminate the connection (See part 2 appendices for the full list of security events).</p> <p>NOTE: This is a critical security event that will need to be queued and sent to CSMS once a connection has been made, as described in use case A04.</p> <p>A security event only needs to be sent once for repeated failed connection attempts, in order to avoid overflow to the offline queue.</p>

4. Use case B Provisioning

4.1. Page 61 - (v1) Requirement B08.FR.19 and N02.FR.15 are ambiguous w.r.t. evse and instance wildcards [676]

Requirement B08.FR.19 tries to catch multiple situations in one requirement, but as a result it is no longer unambiguous. The requirement has therefore been split into multiple requirements, but with the same intention as B08.FR.19.

Delete requirement

ID	Precondition	Requirement definition
B08.FR.19	When Charging Station receives a GetReportRequest with <i>componentVariable</i> elements in which <i>component.instance</i> and/or <i>component.evse</i> are missing	The Charging Station SHALL report for every instance and/or EVSE of the <i>component</i> in <i>componentVariable</i> .

The following new requirements replace B08.FR.19:

New requirements

ID	Precondition	Requirement definition
B08.FR.22	B08.FR.11 AND When Charging Station receives a GetReportRequest with a <i>component</i> in a <i>componentVariable</i> element that has a <i>component.evse.id</i> , but <i>component.evse.connector</i> is missing	The Charging Station SHALL report the component(s) with this <i>component.name</i> , <i>component.instance</i> and <i>component.evse.id</i> for every <i>component.evse.connector</i> , whilst taking into account B08.FR.24.
B08.FR.23	B08.FR.11 AND When Charging Station receives a GetReportRequest with a <i>component</i> in a <i>componentVariable</i> element that has no <i>component.evse.id</i>	The Charging Station SHALL report the component(s) with this <i>component.name</i> , <i>component.instance</i> for every <i>component.evse</i> field (including top level component without <i>component.evse</i>), whilst taking into account B08.FR.24.
B08.FR.24	B08.FR.11 AND When Charging Station receives a GetReportRequest with a <i>component</i> in a <i>componentVariable</i> element that has a value for <i>component.instance</i>	The Charging Station SHALL report the component(s) with this <i>component.name</i> for every <i>component.instance</i> field, whilst taking into account B08.FR.22, B08.FR.23.
B08.FR.25	B08.FR.11 AND When Charging Station receives a GetReportRequest with a <i>component</i> in a <i>componentVariable</i> element that has no <i>component.instance</i> field	The Charging Station SHALL report the component(s) with this <i>component.name</i> for every <i>component.instance</i> field or the component(s) without <i>component.instance</i> field, whichever is the case, whilst taking into account B08.FR.22, B08.FR.23.

4.1.1. Page 319 - N02.FR.15

Exactly the same applies, mutatis mutandis, for requirement N02.FR.15.

Delete requirement

ID	Precondition	Requirement definition
N02.FR.15	When Charging Station receives a GetMonitoringReportRequest with_ <i>componentVariable_</i> elements in which <i>component.instance</i> and/or <i>component.evse</i> are missing	The Charging Station SHALL report for every instance and/or EVSE of the <i>component</i> in <i>componentVariable</i> .

New requirements

ID	Precondition	Requirement definition
N02.FR.18	N02.FR.11 AND When Charging Station receives a GetMonitoringReportRequest with a <i>component</i> in a <i>componentVariable</i> element that has a <i>component.evse.id</i> , but <i>component.evse.connector</i> is missing	The Charging Station SHALL report the component(s) with this <i>component.name</i> , <i>component.instance</i> and <i>component.evse.id</i> for every <i>component.evse.connector</i> , whilst taking into account N02.FR.20.
N02.FR.19	N02.FR.11 AND When Charging Station receives a GetMonitoringReportRequest with a <i>component</i> in a <i>componentVariable</i> element that has no <i>component.evse.id</i>	The Charging Station SHALL report the component(s) with this <i>component.name</i> , <i>component.instance</i> for every <i>component.evse</i> field (including top level component without <i>component.evse</i>), whilst taking into account N02.FR.20.
N02.FR.20	N02.FR.11 AND When Charging Station receives a GetMonitoringReportRequest with a <i>component</i> in a <i>componentVariable</i> element that has a value for <i>component.instance</i>	The Charging Station SHALL report the component(s) with this <i>component.name</i> for every <i>component.instance</i> field, whilst taking into account N02.FR.18, N02.FR.19.
N02.FR.21	N02.FR.11 AND When Charging Station receives a GetMonitoringReportRequest with a <i>component</i> in a <i>componentVariable</i> element that has no <i>component.instance</i> field	The Charging Station SHALL report the component(s) with this <i>component.name</i> for every <i>component.instance</i> field or the component(s) without <i>component.instance</i> field, whichever is the case, whilst taking into account N02.FR.18, N02.FR.19.

4.2. Page 62 - (v1) Use case B09/B10: Extended scenario description [683]

Use case B09 describes the setting of a NetworkConnectionProfile and use case B10 describes how to use NetworkConnectionProfiles to migrate a Charging Station to a new CSMS. The relationship with the variable OCPPCommCtrlr.NetworkConfigurationPriority was not made explicit. The use case descriptions have been updated for that.

4.2.1. Use case B09

The text marked in bold has been added to the scenario description.

No.	Type	Description
1	Name	Setting a new NetworkConnectionProfile.
2	ID	B09
	Functional block	B. Provisioning
3	Objectives	To enable the CSMS to update the connection details on the Charging Station.
4	Description	The CSMS updates the connection details on the Charging Station. For instance in preparation of a migration to a new CSMS. After completion of this use case, the Charging Station to CSMS connection data has been updated.
	Actors	Charging Station, CSMS
	Scenario description	<p>A Charging Station supports at least two network configuration slots, that are identified by a number. The available slot numbers are reported by the Charging Station in the valuesList of variable NetworkConfigurationPriority.</p> <p>For example: <i>valuesList = "0,1,2"</i> in the base report tells CSMS that three configuration slots, numbered 0, 1 and 2, are available (but not necessarily set).</p> <p>The configuration slot number that is used for the active connection is reported by variable OCPPCommCtrlr.ActiveNetworkProfile.</p> <ol style="list-style-type: none"> 1. The CSMS sends a SetNetworkProfileRequest PDU containing an updated connection profile and a <i>configurationSlot</i> out of the <i>valuesList</i> of NetworkConfigurationPriority. 2. The Charging Station receives the PDU, validates the content and stores the new data 3. The Charging Station responds by sending a SetNetworkProfileResponse PDU, with status Accepted
5	Prerequisites	The data supplied by the CSMS matches the Charging Station's capabilities
6	Postcondition(s)	The Charging Station was able to store the new connection data

4.2.2. Requirement for configuration slots

A Charging Station must support at least two configuration slots for network profiles in order to support a migration. The number of the configuration slot must match an entry in the *valuesList* of the [NetworkConfigurationPriority](#).

This was already implicitly required, or else the use case B09 and B10 would not work. It is now made explicit in the following new requirements.

New requirements

ID	Precondition	Requirement definition
B09.FR.05	When the value of <i>configurationSlot</i> in SetNetworkProfileRequest does not match an entry in <i>valuesList</i> of NetworkConfigurationPriority	The Charging Station SHALL respond by sending a SetNetworkProfileResponse message with status <i>Rejected</i>
B09.FR.06		A Charging Station SHALL support at least two configuration slots for network connection profiles.

4.2.3. Use case B10

The text marked in bold has been added to the scenario description.

No.	Type	Description
1	Name	Migrate to new CSMS, using a different NetworkConnectionProfile.
2	ID	B10
	<i>Functional block</i>	B. Provisioning
3	Objectives	After completion of this use case, the Charging Station connects to a new CSMS.
4	Description	This use case describes how a Charging Station can be instructed to connect to a new CSMS, by changing the order of NetworkConnectionProfiles in NetworkConfigurationPriority .
	Actors	Charging Station, CSMS 1, CSMS 2
	Scenario description	<p>A Charging Station supports at least two network configuration slots, that are identified by a number. The available slot numbers are reported by the Charging Station in the <i>valuesList</i> of variable NetworkConfigurationPriority.</p> <p>For example: <i>valuesList</i> = "0,1,2" in the base report tells CSMS that three configuration slots, numbered 0, 1 and 2, are available (but not necessarily set).</p> <p>The <i>value</i> of NetworkConfigurationPriority reports the order in which network configurations are tried to make a connection.</p> <p>For example: <i>value</i> = "1,0" means that Charging Station will first try configuration slot 1 and if that fails after the number of attempts configured in NetworkProfileConnectionAttempts, it will try to connect with configuration slot 0.</p> <p>1. CSMS 1 sets a new value for the NetworkConfigurationPriority configuration variable via SetVariablesRequest, such that the NetworkConnectionProfile for CSMS 2 becomes first in the list and the existing connection to CSMS 1 becomes second in the list. 2. The Charging Station responds with a SetVariablesResponse with status <i>Accepted</i> 3. CSMS 1 instructs the Charging Station to perform a Reset OnIdle. 4. The Charging Station reboots and connects via the new primary NetworkConnectionProfile to CSMS 2.</p>
5	Prerequisites	Use case B09 - Setting a new NetworkConnectionProfile was executed successfully prior to this use case The data supplied by the CSMS matches the Charging Station's capabilities
6	Postcondition(s)	The Charging Station is connected via a different NetworkConnectionProfile .

5. Use case C Authorization

5.1. Page 90 - (v1) C07 requirements for certificateStatus missing [680]

In case of ISO 15118 Plug&Charge the AuthorizeResponse returns a certificateStatus of type AuthorizeCertificateStatusEnumType. Requirement C07.FR.04 states that an authorization status must be returned, but the exact values are not defined.

Requirements have been added that describe which values to use for certificateStatus.

New requirements

ID	Precondition	Requirement definition	Note
C07.FR.13	C07.FR.04 AND the certificate chain (provided in certificate or iso15118CertificateHashData) is valid AND authorization status of idToken is one of Blocked, Expired, Invalid, Unknown	CSMS SHALL return an AuthorizationResponse containing a certificateStatus = ContractCancelled and the authorization status in idTokenInfo.status.	Certificate is valid, but EMAID is not accepted.
C07.FR.14	C07.FR.04 AND the certificate chain (provided in certificate or iso15118CertificateHashData) is valid AND authorization status of idToken is NOT one of Blocked, Expired, Invalid, Unknown	CSMS SHALL return an AuthorizationResponse containing a certificateStatus = Accepted and the authorization status in idTokenInfo.status.	Charging can still not be allowed if idTokenInfo.status is other than Accepted (e.g. ConcurrentTx or NotAtThisLocation).
C07.FR.15	C07.FR.04 AND the certificate chain (provided in certificate or iso15118CertificateHashData) has expired	CSMS SHALL return an AuthorizationResponse containing a certificateStatus = CertificateExpired and an idTokenInfo.status = Expired	If certificate is expired, then status of idToken is also reported expired.
C07.FR.16	C07.FR.04 AND the certificate chain (provided in certificate or iso15118CertificateHashData) has been revoked	CSMS SHALL return an AuthorizationResponse containing a certificateStatus = CertificateRevoked and an idTokenInfo.status = Invalid	If certificate is revoked, then status of idToken is reported as invalid.
C07.FR.17	C07.FR.04 AND the certificate chain (provided in certificate or iso15118CertificateHashData) cannot be verified or is invalid	CSMS SHALL return an AuthorizationResponse containing a certificateStatus = CertChainError and an idTokenInfo.status = Invalid	If certificate is cannot be verified, then status of idToken is reported as invalid.

5.1.1. Page 408 - AuthorizeCertificateStatusEnumType

The enumeration AuthorizeCertificateStatusEnumType contains some values that are not used. These enumeration values continue to exist, so as not to change the JSON schema, but their description is changed to show that these values have no meaning.

Updated text in bold:

AuthorizeCertificateStatusEnumType

Value	Description
Accepted	Positive response
SignatureError	<not used>
CertificateExpired	If the contract certificate in the AuthorizeRequest is expired.
CertificateRevoked	If the Charging Station or CSMS determine (via a CRL or OCSP response) that the contract certificate in the AuthorizeRequest is marked as revoked.
NoCertificateAvailable	<not used>
CertChainError	If the contract certificate contained in the AuthorizeRequest message is not valid.
ContractCancelled	If the EMAID provided by EVCC is invalid, unknown, expired or blocked.

5.2. Page 97 - (v1) Requirement C10.FR.06 needs to be removed [685]

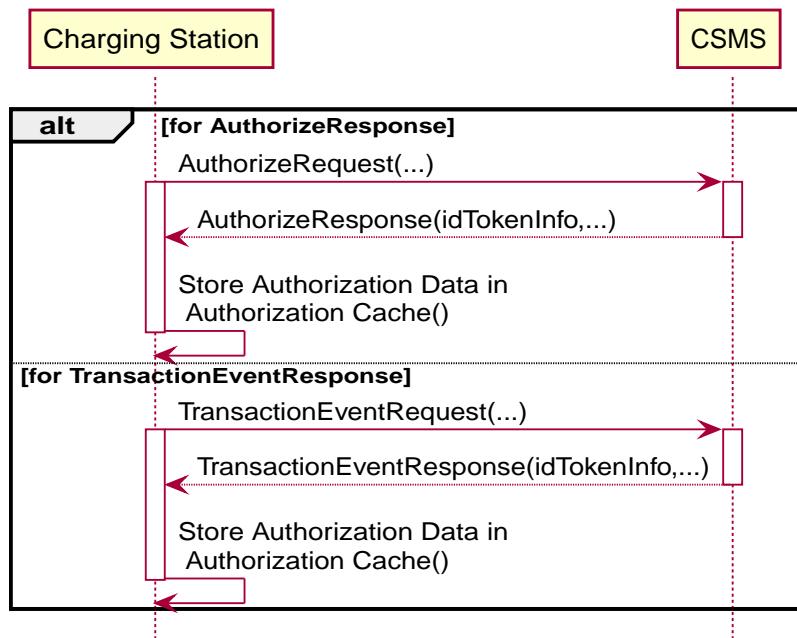
Requirement C10.FR.06 is an invalid requirement, because the ReserveNowRequest does not contain [IdTokenInfo](#), so there is no information to update the Authorization Cache with.

Deleted requirement

ID	Precondition	Requirement definition	Note
C10.FR.06	Upon receipt of ReserveNowRequest .	The Charging Station SHALL update the Authorisation Cache entry.	The update is to be done with the IdTokenInfo value from the request as described under Authorization Cache .

5.2.1. Page 96 - Update sequence diagram

As a result of the above, the "for ReserveNowRequest" part has been removed from sequence diagram "Figure 31".



5.3. Page 102 - (v1) Requirement C13.FR.04 enhanced [701]

Requirement C13.FR.04 suggests that any identifier must be accepted, but that was not the intention. In fact, it is in conflict with use case C15 that describes offline authorization of an unknown identifier. Requirement C15.FR.08 says that any [unknown](#) identifier not in Authorization Cache or Local Authorization List (prerequisite of C15) must be accepted. C13.FR.04 is updated to reflect this.

Changed requirement

	ID	Precondition	Requirement definition	Note
Old text	C13.FR.04	If configuration variable OfflineTxForUnknownIdEnable is true AND The Charging Station is offline.	Any identifier SHALL be allowed to authorize a transaction.	
New text	C13.FR.04	If configuration variable OfflineTxForUnknownIdEnable is true AND The Charging Station is offline.	Any identifier that is present in neither the Authorization Cache nor the Local Authorization List SHALL be allowed to authorize a transaction.	See also C15.FR.08

6. Use Case E Transactions

6.1. Page 147 - (v1) Use case E07 - Scenario description step order incorrect [704]

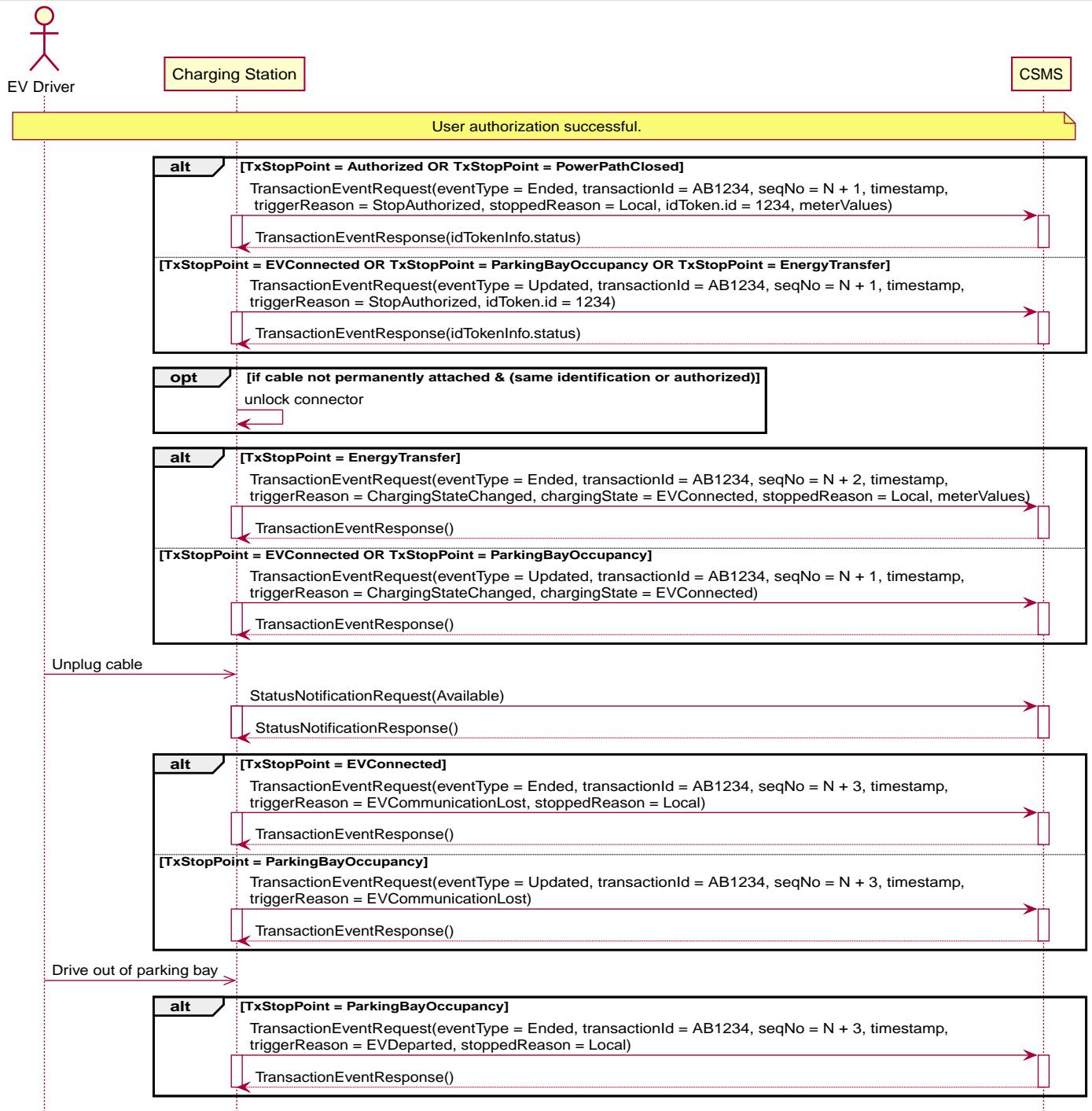
The Charging Station must first stop the energy transfer as described by step 4, before transmitting the TransactionEventRequest(eventType = Ended) message from step 2 and 3.

No.	Type	Description
Old text	<i>Scenario description</i> TxStopPoint = Authorized (or PowerPathClosed)	<ol style="list-style-type: none">1. The EV Driver presents IdToken a second time to end charging.2. The Charging Station sends a TransactionEventRequest (eventType = Ended) with triggerReason = StopAuthorized and stoppedReason = Local.3. The CSMS responds with a TransactionEventResponse.4. The Charging Station stops the energy transfer and if the cable is not permanently attached, the Charging Station unlocks the cable.
New text	<i>Scenario description</i> TxStopPoint = Authorized (or PowerPathClosed)	<ol style="list-style-type: none">1. The EV Driver presents IdToken a second time to end charging.2. The Charging Station stops the energy transfer and if the cable is not permanently attached, the Charging Station unlocks the cable.3. The Charging Station sends a TransactionEventRequest (eventType = Ended) with triggerReason = StopAuthorized and stoppedReason = Local.4. The CSMS responds with a TransactionEventResponse.

6.2. Page 148 - (v1) Use case E07: Wrong triggerReason shown in sequence diagram fig. 56 [687]

The fourth TransactionEventRequest in sequence diagram Figure 56 contains an incorrect triggerReason and should not have an idToken. Changed to triggerReason = ChargingStateChanged, chargingState = EVConnected.

Figure 56. Sequence Diagram: Transaction locally stopped by IdToken with TransactionEventRequest reported strictly by TxStopPoint configuration



6.3. Page 150 - (v1) Use case E07: Clarify 'normal' and 'correct' for stoppedReason [693]

Some requirements in E07 mention "ended in a normal way" and "set to a correct value", but do not explain what normal and correct is.

	ID	Precondition	Requirement definition	Note
Old text	E07.FR.04	If a transaction is ended in a normal way.	The stoppedReason element MAY be omitted.	e.g. EV-driver presented IdToken to stop the transaction.
New text	E07.FR.04	If a transaction is stopped on request of the EV driver at the Charging Station.	Charging Station MAY omit the stoppedReason element from the final TransactionEventRequest with eventType = Ended	e.g. EV-driver presented IdToken to stop the transaction or pressed a "stop" button (not the emergency stop). See use case F03 for remotely stopping.

	ID	Precondition	Requirement definition	Note
Old text	E07.FR.05	If a transaction is ended in a normal way	The <code>stoppedReason</code> SHOULD be assumed 'Local'.	e.g. EV-driver presented IdToken to stop the transaction.
New text	E07.FR.05	If a transaction is stopped on request of the EV driver at the Charging Station.	Charging Station SHOULD use a <code>stoppedReason</code> = Local in the final TransactionEventRequest with <code>eventType</code> = Ended.	e.g. EV-driver presented IdToken to stop the transaction or pressed a "stop" button (not the emergency stop). See use case F03 for remotely stopping.
Old text	E07.FR.06	If the transaction is not ended normally.	<code>stoppedReason</code> SHOULD be set to a correct value.	
New text	E07.FR.06	If a transaction is stopped, but not on request of the EV driver at the Charging Station.	Charging Station SHOULD use the most appropriate value from <code>ReasonEnumType</code> for <code>stoppedReason</code> in the final TransactionEventRequest with <code>eventType</code> = Ended.	Apart from remotely stopping (Remote), CSMS revoking authorization (DeAuthorized) or disconnecting the EV (EVDisconnected), most other reasons are related to technical faults or energy limitations.

6.3.1. Page 403 - TransactionType field `stoppedReason`

The description for field `stoppedReason` in TransactionEventRequest has been improved to make clear that this event does not have to concur with the TransactionEventRequest(Ended) or TxStopPoint, but may have happened some time before.

TransactionType

	Field Name	Field Type	Card.	Description
Old text	<code>stoppedReason</code>	<code>ReasonEnumType</code>	0..1	Optional. This contains the reason why the transaction was stopped. MAY only be omitted when Reason is "Local".
New text	<code>stoppedReason</code>	<code>ReasonEnumType</code>	0..1	Optional. The <code>stoppedReason</code> is the reason/event that initiated the process of stopping the transaction. It will normally be the user stopping authorization via card (Local or MasterPass) or app (Remote), but it can also be CSMS revoking authorization (DeAuthorized), or disconnecting the EV when TxStopPoint = EVConnected (EVDisconnected). Most other reasons are related to technical faults or energy limitations. MAY only be omitted when <code>stoppedReason</code> is "Local"

7. Use Case F Remote Control

7.1. Page 180 - (v1) Requirement F03.FR.03 contains wrong precondition [700]

The precondition of requirement F03.FR.03 was incorrectly merged from Errata v2 into Edition 2, and the associated Note was not relevant for this situation.

It needs to be changed as follows:

Changed requirement

	ID	Precondition	Requirement definition	Note
Old text	F03.FR.03	F03.FR.01 AND TxStopPoint configuration causes transaction to end (E.g. TxStopPoint is NOT Authorized or PowerPathClosed)	The Charging Station SHALL send a TransactionEventRequest (<i>eventType</i> = Ended, <i>triggerReason</i> = RemoteStop, <i>stoppedReason</i> = Remote) to the CSMS.	For example when TxStopPoint = EVConnected and EV is disconnected after the RequestStopTransactionRequest.
New text	F03.FR.03	F03.FR.01 AND TxStopPoint configuration causes transaction to end (E.g. TxStopPoint is NOT Authorized or PowerPathClosed)	The Charging Station SHALL send a TransactionEventRequest (<i>eventType</i> = Ended, <i>triggerReason</i> = RemoteStop, <i>stoppedReason</i> = Remote) to the CSMS.	For example when TxStopPoint = EVConnected and EV is disconnected after the RequestStopTransactionRequest.

7.2. Page 187 - (v1) Requirement F06.FR.12 is too strict [707]

Requirement F06.FR.12 explicitly tells a Charging Station to reject a TriggerMessageRequest for a *requestedMessageStatusNotification* without *evse* or *evse.connectorId*. There is no need to require this from a Charging Station, since F06.FR.13 already mandates that CSMS shall provide an *evse.connectorId* (and an *evse.id*, because that is mandatory in the *evse* object) in this message.

The requirement definition of F06.FR.12 has been relaxed from SHALL to a MAY, so that a Charging Station implementation that is able to handle to a request without *evse.connectorId* and an implementation that rejects this, are both allowed, since a CSMS is not allowed to send a request without *evse.connectorId*.

Changed requirement

	ID	Precondition	Requirement definition	Note
Old text	F06.FR.12	If a Charging Station receives a TriggerMessageRequest with <i>requestedMessage</i> set to: <i>StatusNotification</i> AND (<i>evse</i> is omitted OR <i>evse.connectorId</i> is omitted)	The Charging Station SHALL respond with a TriggerMessageResponse with status Rejected .	StatusNotification messages can only be sent at connector level.
New text	F06.FR.12	If a Charging Station receives a TriggerMessageRequest with <i>requestedMessage</i> set to: <i>StatusNotification</i> AND (<i>evse</i> is omitted OR <i>evse.connectorId</i> is omitted)	The Charging Station MAY respond with a TriggerMessageResponse with status Rejected .	StatusNotification messages can only be requested at connector level.

8. Use Case G Availability

8.1. Page 192 - (v1) G01.FR.08 contradicts H01.FR.24 [692]

Requirement G01.FR.08 states that a *StatusNotification* must be sent when a connector becomes reserved. However, this topic is already covered in use case H01 in a slightly different way. Therefore, the "becomes reserved" must be removed from G01.FR.08 and left to H01.FR.24.

Table 1. G01 - Requirements

	ID	Precondition	Requirement definition
Old text	G01.FR.08	When a connector of an EVSE becomes reserved or a cable is plugged-in AND The EVSE has multiple connectors	The Charging Station SHOULD NOT send a StatusNotificationRequest for the other connector(s), even though they are no longer usable.
New text	G01.FR.08	When a cable is plugged in to a connector of an EVSE AND The EVSE has multiple connectors	The Charging Station SHOULD NOT send a StatusNotificationRequest for the other connector(s), even though they are no longer usable.

9. Use Case H Reservation

9.1. Page 205 - (v1) Missing option to send NotifyEvent instead of StatusNotification [699]

Instead of StatusNotificationRequest it is also allowed to send a NotifyEvent(AvailabilityState) for the connector, which will become the preferred method in future OCPP releases. This option was missing from use case H and is added to the following requirements.

Changed requirements

	ID	Precondition	Requirement definition	Note
Old text	H01.FR.20	H01.FR.04 AND amount of EVSEs available equals the amount of reservations	The Charging Station SHALL send a StatusNotificationRequest with connectorStatus = Reserved for all connectors of the EVSE.	If an EVSE is reserved, all of its connectors are reported as reserved.
New text	H01.FR.20	H01.FR.04 AND amount of EVSEs available equals the amount of reservations	The Charging Station SHALL send for all connectors of the EVSE: - a StatusNotificationRequest with connectorStatus = Reserved , OR - a NotifyEventRequest with component = "Connector" , variable = "AvailabilityState" , trigger = "Delta" , actualValue = "Reserved"	If an EVSE is reserved, all of its connectors are reported as reserved.
Old text	H01.FR.23	If the Charging Station receives a ReserveNowRequest for evsel/d AND this EVSE is <i>Available</i>	The Charging Station SHALL respond with a ReserveNowResponse with status Accepted AND SHALL send a StatusNotificationRequest with connectorStatus = Reserved for all connectors of the EVSE.	If an EVSE is reserved, all of its connectors are reported as reserved.
New text	H01.FR.23	If the Charging Station receives a ReserveNowRequest for evsel/d AND this EVSE is <i>Available</i>	The Charging Station SHALL respond with a ReserveNowResponse with status Accepted AND SHALL send for all connectors of the EVSE: - a StatusNotificationRequest with connectorStatus = Reserved , OR - a NotifyEventRequest with component = "Connector" , variable = "AvailabilityState" , trigger = "Delta" , actualValue = "Reserved"	If an EVSE is reserved, all of its connectors are reported as reserved.
Old text	H01.FR.24	H01.FR.06 AND amount of reservations for a specific connectorType equals the amount of available EVSEs with that specific connectorType	The Charging Station SHALL send a StatusNotificationRequest with connectorStatus = Reserved for all connectors of the EVSEs with the specific connectorType .	If an EVSE is reserved for a specific connectorType , all connectors on the EVSE are reported as reserved.
New text	H01.FR.24	H01.FR.06 AND amount of reservations for a specific connectorType equals the amount of available EVSEs with that specific connectorType	The Charging Station SHALL send for all connectors of the EVSEs that have the specific connectorType - a StatusNotificationRequest with connectorStatus = Reserved , OR - a NotifyEventRequest with component = "Connector" , variable = "AvailabilityState" , trigger = "Delta" , actualValue = "Reserved"	If an EVSE is reserved for a specific connectorType , all connectors on the EVSE are reported as reserved.

9.1.1. Page 203 - (v1) Added option to use case description to send NotifyEventRequests

Use case H01 scenario S2 only mentions StatusNotificationRequests, but the use of NotifyEventRequests is also allowed. This has been added in **bold**, similarly to how this was done in use case G01 StatusNotification.

S2	Scenario objective	Reserve a specific EVSE at a Charging Station
----	--------------------	---

	<i>Scenario description</i>	<ol style="list-style-type: none"> 1. EV Driver asks the CSMS to reserve a specific EVSE at the Charging Station. 2. The CSMS sends ReserveNowRequest with a EVSE to a Charging Station. 3. Upon receipt of ReserveNowRequest, the Charging Station responds with ReserveNowResponse with status Accepted. 4. The Charging Station sends StatusNotificationRequest with the status Reserved for all Connectors of that EVSE. 5. The CSMS responds with StatusNotificationResponse to the Charging Station.
	<i>Alternative scenario description</i>	<p>Steps 1, 2 and 3 as above.</p> <p>4. Instead of a StatusNotificationRequest a Charging Station can send a NotifyEventRequest with <i>trigger</i> = <i>Delta</i> for <i>component.name</i> = "Connector" and the EVSE number in <i>evse.id</i> and the connector number in <i>evse.connectorId</i>, <i>variable</i> = "AvailabilityState" and <i>actualValue</i> = "Reserved".</p> <p>5a. Optionally, Charging Station can also report a NotifyEventRequest for <i>component</i> = "EVSE", <i>variable</i> = "AvailabilityState" and <i>actualValue</i> = "Reserved", and when applicable, also report this for <i>component</i> = "ChargingStation".</p>
	<i>Prerequisite(s)</i>	The specified EVSE of the Charging Station has status <i>Available</i>
	<i>Postcondition(s)</i>	<p>Successful postcondition:</p> <p>The Charging Station has accepted the ReserveNowRequest AND sent StatusNotificationRequests with status <i>Reserved</i>.</p> <p>Failure postcondition:</p> <p>The Charging Station has rejected the ReserveNowRequest OR The Charging Station has NOT sent StatusNotificationRequests with status <i>Reserved</i>.</p>

9.2. Page 209 - (v1) Remark about authorization in use case H03 [711]

Use case H01 has a remark that says: "It is RECOMMENDED to validate the Identifier with an AuthorizeRequest after reception of ReserveNowRequest and before the start of the transaction." Use case H03 about using a reservation does not have a recommendation to validate before starting the transaction.

In order to be consistent with H01, this has been added to the remark of H03, as shown in **bold**:

7	Error handling	n/a
8	Remark(s)	It is RECOMMENDED to validate the Identifier with an AuthorizeRequest after reception of ReserveNowRequest and before the start of the transaction.

9.3. Page 210 - (v1) Requirement H03.FR.08 is not clear about groupIdToken lookup [684]

Requirement H03.FR.08 can mistakenly be interpreted as having to look up the **groupIdToken** in the Local Authorization List or Authorization Cache. However, the intention is to look up the incoming *idToken* to get its associated *groupIdToken*, if any.

The requirements H03.FR.07 and H03.FR.08 exist to make clear, that for a reserved EVSE or connector a lookup or authorize request for *idToken* is needed when a **groupIdToken** is involved.

Changed requirement

	ID	Precondition	Requirement definition
Old text	H03.FR.08	H03.FR.07 AND If it is not found in the Local Authorization List or Authorization Cache.	The Charging Station SHALL send an AuthorizeRequest for the incoming <i>IdToken</i> to the CSMS in order to get its associated <i>groupIdToken</i> .
New text	H03.FR.08	H03.FR.07 AND If the incoming <i>IdToken</i> is not found in the Local Authorization List or Authorization Cache.	The Charging Station SHALL send an AuthorizeRequest for the incoming <i>IdToken</i> to the CSMS in order to get its associated <i>groupIdToken</i> . (Note: This AuthorizeRequest may already have been performed when the <i>idToken</i> was presented for authorization.)

10. Use Case J Meter Values

10.1. Page 228 - (v1) Requirement J01.FR.14 is unclear that meter values for all EVSEs must be sent [674]

J01 is not clear about the fact that MeterValuesRequest for clock-aligned data always need to be sent for all locations, including the grid energy meter, which is designated by `evselid = 0`. It is stated in the text in par. 2.3: "When a Charging Station can measure the same measurand on multiple locations or phases, all possible locations and/or phases SHALL be reported when configured in one of the relevant Configuration Variables." The requirement J01.FR.14 has been extended to refer to all possible locations and phases.

Changed requirement

	ID	Precondition	Requirement definition	Note
Old text	J01.FR.14	When configured to send MeterValuesRequest , See: Meter Values - Configuration	The Charging Station SHALL send MeterValuesRequest messages to the CSMS as configured.	
New text	J01.FR.14	When configured to send MeterValuesRequest , See: Meter Values - Configuration	The Charging Station SHALL send MeterValuesRequest messages to the CSMS as configured in Meter Values - Configuration , for all evselids, locations and phases for which a configured measurand is supported .	It is possible that certain measurands are not available for every location. For example, <code>evselid = 0</code> (grid meter) will not have a "Current.Offered" or "SoC" measurand.

10.2. Page 230 - (v1) Requirement J02.FR.10 refers to all TransactionEventRequest messages, but should be specific to only eventType = Updated [705]

A TransactionEventRequest(Started/Update) should only have sampled values that are part of the same sampling interval. Ideally, this would mean that all sampled values have the same timestamp, and can thus be part of a single `meterValue` element. In practice, however, when multiple measurands or meters are sampled the associated timestamps may differ slightly. This is acceptable, as long as the samples belong to the same sampling interval.

This was the intention of J02.FR.10 with the phrase "belong to the timestamp in the message", but it could also be interpreted as requiring identical timestamps. Also, it forgot to mention that it only applies to Started and Updated events, since an Ended event can contain `metervalues` for multiple timestamps.

Changed requirement

	ID	Precondition	Requirement definition	Note
Old text	J02.FR.10		The <code>meterValue</code> measurements in the same TransactionEventRequest message SHALL all belong to the timestamp in the message	<code>meterValues</code> for other timestamps should be sent in separate TransactionEventRequest messages.
New text	J02.FR.10	If a TransactionEventRequest message with <code>eventType = Started</code> or <code>eventType = Update</code> contains multiple <code>meterValue</code> elements, rather than one <code>meterValue</code> with one or more <code>sampledValue</code> elements	All <code>meterValue</code> elements SHALL have a timestamp that is within the current sampling interval, i.e.: $(\text{transaction event timestamp} - \text{SampledDataTxUpdatedInterval}) < \text{meterValue.timestamp} \leq \text{transaction event timestamp}$	Only for <code>eventType = Ended</code> can a TransactionEventRequest have <code>meter values</code> for multiple intervals.

10.3. Page 231 - (v1) J01 misses requirement that meter value must be for current transaction [673]

It is perhaps obvious, but not stated anywhere. Transaction-related meter values reported in the TransactionEventRequest must only report the measurand(s) associated with the evse of the TransactionEventRequest.

New requirement

ID	Precondition	Requirement definition	Note
J02.FR.22		Meter values reported in a TransactionEventRequest message SHALL all be related to EVSE on which the transaction is taking place.	

11. Use Case K Smart Charging

11.1. Page 238 - (v1) Text in section 3.3 does not match ChargingProfileKindEnumType description [708]

The description of the ChargingProfileKindEnumType Relative was updated in Edition 2 to be more exact. This update was unfortunately not performed in section 3.3 Charging Profile Recurrency that introduces the charging profile kinds.

Below is the updated text shown in bold:

	ChargingProfile Kind	Description
Old text	Relative	Charging schedule periods start when ChargingProfile is activated. In most cases this will be at start of the power delivery. When a ChargingProfile is received for a transaction in progress, then it should activate immediately. No value for <i>startSchedule</i> should be supplied.
New text	Relative	Charging schedule periods should start when the EVSE is ready to deliver energy. i.e. when the EV driver is authorized and the EV is connected. When a ChargingProfile is received for a transaction that is already charging, then the charging schedule periods should remain relative to the PowerPathClosed moment. No value for <i>startSchedule</i> should be supplied.

12. Use Case L FirmwareManagement

12.1. Page 287 - (v1) Improved title of figure 119 [695]

Figure 119 shows the transitions between all [FirmwareStatusEnumType](#) values. As such, it is a state transition diagram. The title, however, calls it "Firmware update process", which is not correct, because it does not cover all steps for performing a firmware update.

Old text	Figure 119. Firmware update process
New text	Figure 119. Firmware status transitions

13. Use Case M ISO 15118 CertificateManagement

13.1. Page 310 - (v1) M04.FR.07 has an incorrect requirement definition [703]

Requirement M04.FR.07 mentions a hash algorithm used during installation, but no hash algorithm is used to install a certificate. The intention of this requirement was, as is suggested by the note, that the CSMS, when deleting a certificate, uses the same *hashAlgorithm* as the Charging Station when generating the *certificateHashData* for a certificate.

	ID	Precondition	Requirement definition	Note
Old text	M04.FR.07	When deleting a certificate	The CSMS SHALL use the <i>hashAlgorithm</i> , which was used to install the certificate.	When a new firmware is installed it is RECOMMENDED that the CSMS requests the certificate first using GetInstalledCertificateIdsRequest to be sure of the used <i>hashAlgorithm</i> .
New text	M04.FR.07	When deleting a certificate	The CSMS SHALL use the same <i>hashAlgorithm</i> as the Charging Station uses to report the certificateHashData for the certificate in the GetInstalledCertificateIdsResponse.	This ensures CSMS uses a <i>hashAlgorithm</i> that is supported by the Charging Station.

14. Use Case N Diagnostics

14.1. Page 317 - (v1) N01.FR.10 not clear when to report UploadFailure [696]

Requirement N01.FR.10 does not make clear whether the LogStatusNotification about failure to upload should be sent after all retry attempts or at each failure. Both options are allowed, but it is recommended to do this after all retry attempts have failed. This has been added to the note.

	ID	Precondition	Requirement definition	Note
Old text	N01.FR.10	When uploading a log document failed	The Charging Station SHALL send a LogStatusNotificationRequest with status <i>UploadFailure</i> , <i>BadMessage</i> , <i>PermissionDenied</i> OR <i>NotSupportedOperation</i> .	It is RECOMMENDED to send a status that describes the reason of failure as precise as possible.
New text	N01.FR.10	When uploading a log document failed	The Charging Station SHALL send a LogStatusNotificationRequest with status <i>UploadFailure</i> , <i>BadMessage</i> , <i>PermissionDenied</i> OR <i>NotSupportedOperation</i> .	It is RECOMMENDED to send the status only after all retry attempts have failed . A Charging Station MAY send a new Uploading status upon each retry attempt.

14.2. Page 331 - (v1) Requirement N09.FR.04 has been rephrased [688]

Requirement N09.FR.04 for CSMS states that a reference to a customer by either *idToken*, *customerCertificate* or *customerIdentifier* is needed, but it does not tell what to do if that is not obeyed.

A new requirement has been added for Charging Station for this case.

New requirement

ID	Precondition	Requirement definition	Note
N09.FR.09	When CustomerInformationRequest contains none of <i>idToken</i> , <i>customerCertificate</i> or <i>customerIdentifier</i> OR CustomerInformationRequest contains more than one of <i>idToken</i> , <i>customerCertificate</i> or <i>customerIdentifier</i>	Charging Station SHALL respond with status = Invalid	Only one value for either <i>idToken</i> , <i>customerCertificate</i> or <i>customerIdentifier</i> may be provided. Charging Station counterpart requirement of N09.FR.04.

15. Messages

15.1. Page 353 - (v1) Clarification for use of certificate and iso15118CertificateHashData in AuthorizeRequest [675]

In case of ISO 15118 Plug&Charge the AuthorizeRequest has two optional fields: *certificate* and *iso15118CertificateHashData*. The behaviour is described in requirements C07.FR.05 and C07.FR.06, but it was not clear enough that only one of these fields is needed.

The field *certificate* contains the entire contract certificate chain. It is only needed in case of central contract validation, where Charging Station cannot locally validate the contract certificate, e.g. because it is lacking the root certificate. If *certificate* is provided, it is no longer needed to provide *iso15118CertificateHashData*.

Text in **bold** is added to the description.

AuthorizeRequest

Field Name	Field Type	Card.	Description
certificate	string[0..5500]	0..1	Optional. The X.509 certificate chain presented by EV and encoded in PEM format. Order of certificates in chain is from leaf up to (but excluding) root certificate. Only needed in case of central contract validation when Charging Station cannot validate the contract certificate.
idToken	IdTokenType	1..1	Required. This contains the identifier that needs to be authorized.
iso15118CertificateHashData	OCSPRequestDataType	0..4	Optional. Contains the information needed to verify the EV Contract Certificate via OCSP. Not needed if certificate is provided.

15.2. Page 381 - (v1) Updated description for idToken in TransactionEventRequest [709]

The *idToken* in a TransactionEventRequest is only supposed to be sent after an id token has been authorized, either locally or centrally. This happens when starting and stopping the authorization for a transaction. CSMS then returns the validity status of the *idToken* in the TransactionRequestResponse. When a transaction is stopped via a RequestStopTransactionRequest or a ResetRequest, no id token is involved and as a result no *idToken* should be provided in the TransactionEventRequest, because CSMS does not need to check validity.

The description of *idToken* has been updated to make this clear.

	Field Name	Field Type	Card.	Description
Old text	idToken	IdTokenType	0..1	Optional. This contains the identifier for which a transaction is (or will be) started or stopped. Is required when the EV Driver becomes authorized for this transaction and when the EV Driver ends authorization. The IdToken should only be sent once in a TransactionEventRequest for every authorization (for starting or for stopping) done for this transaction.
New text	idToken	IdTokenType	0..1	Optional. This contains the identifier for which a transaction is (or will be) started or stopped. Is required when the EV Driver becomes authorized for this transaction and when the EV Driver ends authorization. The IdToken should only be sent once in a TransactionEventRequest for every authorization (for starting or for stopping) done for this transaction, so that CSMS can return the <i>idTokenInfo</i> in the TransactionEventResponse. <i>idToken</i> should not be present in the TransactionEventRequest when a transaction is ended by a RequestStopTransactionRequest or a ResetRequest.

16. Data Types

16.1. Page 386 - (v1) issuerKeyHash in CertificateHashDataType must be type identifierString [691]

The field type of *issuerKeyHash* in CertificateHashDataType must be "identifierString[0..128]", instead of "string[0..128]". The difference is, that identifierString is case-insensitive. This is, however, not checked by the JSON schema, and as a result this change does not affect the JSON schema.

Changed field type for issuerKeyHash:

CertificateHashDataType

Field Name	Field Type	Card.	Description
hashAlgorithm	HashAlgorithmEnumType	1..1	Required. Used algorithms for the hashes provided.
issuerNameHash	identifierString[0..128]	1..1	Required. The hash of the issuer's distinguished name (DN), that must be calculated over the DER encoding of the issuer's name field in the certificate being checked.
issuerKeyHash	identifierString[0..128]	1..1	Required. The hash of the DER encoded public key: the value (excluding tag and length) of the subject public key field in the issuer's certificate.
serialNumber	identifierString[0..40]	1..1	Required. The string representation of the hexadecimal value of the serial number without the prefix "0x" and without leading zeroes.

16.2. Page 396 - (v1) NetworkConnectionProfileType [683]

The data type NetworkConnectionProfileType has two fields that do not serve a purpose.

- The field *ocppVersion* has no use, because the selection of the OCPP version that a charging station will use, is done during the websocket handshake. It is not determined by the NetworkConnectionProfile.
- The field *ocppInterface* is mandatory, but in most cases a CSMS will not even be aware of which interfaces a charging station supports or should use to connect. It is a mandatory field, so CSMS must provide something, but that might not match with the capability of the charging station. To remedy this, a charging station is allowed to use a different interface if it cannot connect via the given *ocppInterface*.

The descriptions of these fields have been updated with text in bold to make this clear.

Changed descriptions in NetworkConnectionProfileType

Field Name	Field Type	Card.	Description
ocppVersion	OCPPVersionEnumType	1..1	Required. Defines the OCPP version used for this communication function. This field is ignored, since the OCPP version to use is determined during the websocket handshake.
...			
ocppInterface	OCPPInterfaceEnumType	1..1	Required. Applicable Network Interface. Charging Station is allowed to use a different network interface to connect if the given one does not work.
...			

17. Enumerations

17.1. Page 419 - (v1) Description for idTokenEnumType MacAddress [664]

A description is missing for value MacAddress of IdTokenEnumType.

Value	Description
MacAddress	The MacAddress of the EVCC (Electric Vehicle Communication Controller) that is connected to the EVSE. This is used as a token type when the MAC address is used for authorization ("Autocharge").

18. Referenced Components and Variables

18.1. Page 436 - (v1) Websocket-related variables in Part 4 [690]

Add the following note below section heading "General":

NOTE WebSocket-related variables are described in "[OCPP-2.0.1 Part 4 JSON over WebSockets](#)".

18.1.1. Page 430 - 2.1.13 WebSocketPingInterval

This configuration variable at this location has "Required = No", but that is confusing, because it is required for a WebSocket implementation. All WebSocket configuration variables are described in Part 4.

Replace table describing this variable with a reference to Part 4, as follows:

This configuration variable is described in "[OCPP-2.0.1 Part 4 JSON over WebSockets](#)".

18.2. Page 444 - (v1) SecurityCtrlr.BasicAuthPassword and Identity should have dataType=string

The *dataType* of SecurityCtrlr.BasicAuthPassword is mistakenly shown as "passwordString". The content is similar to a passwordString as defined in part 2, but the device model *dataType* is "string". The same applies to SecurityCtrlr.Identity which shows *dataType* "identifierString".

Replace the descriptions of `BasicAuthPassword` and `Identity` by the updated text below. This change has also been made in Part 2 Appendix chapter 3 "Standardized Components".

Updated dataType:
(change shown in ***bold italic***)

BasicAuthPassword

The basic authentication password is used for HTTP Basic Authentication. The configuration value is write-only, so that it cannot be accidentally stored in plaintext by the CSMS when it reads out all configuration values.

Required	no		
Component	componentName	SecurityCtrlr	
Variable	variableName	BasicAuthPassword	
	variableAttributes	mutability	WriteOnly
	variableCharacteristics	dataType	string
		maxLimit	40 (Max length of the BasicAuthPassword)

Description	The basic authentication password is used for HTTP Basic Authentication. The password SHALL be a randomly chosen passwordString with a sufficiently high entropy, consisting of minimum 16 and maximum 40 characters (alphanumeric characters and the special characters allowed by passwordString). The password SHALL be sent as a UTF-8 encoded string (NOT encoded into octet string or base64). This configuration variable is write-only, so that it cannot be accidentally stored in plaintext by the CSMS when it reads out all configuration variables. This configuration variable is required unless only "security profile 3 - TLS with client side certificates" is implemented.
--------------------	---

*Updated dataType:
(change shown in **bold italic**)*

Identity

Required	no			
Component	componentName	SecurityCtrlr		
Variable	variableName	Identity		
	variableAttributes	mutability	ReadOnly or ReadWrite	
	variableCharacteristics	dataType	string	
		maxLimit	48 (Charging Station Identity)	
Description	The Charging Station identity. Identity is an identifierString , however because this value is also used as the basic authentication username, the colon character ':' SHALL not be used. Maximum length was chosen to ensure compatibility with EVSE ID from [EMI3-BO] "Part 2: business objects".			

18.3. Page 452 - (v1) Incomplete description TxStopPoint Authorized and PowerPathClosed [704]

A transaction shall not end while energy transfer is still ongoing, otherwise it is not possible to report a correct final meter value for the transaction. TxStopPoints Authorized and PowerPathClosed will trigger the transaction to be ended after a StopAuthorized or Deauthorized event, but the Charging Station must wait until the energy transfer has been ended, before transmitting the TransactionEventRequest with eventType = Ended, so that this message can contain the final meter values.

The description of these TxStopPoints has been enhanced to make this clear.

2.6.6.2 TxStopPoint values

Value	Description
Authorized	Driver or EV is no longer authorized, this can also be some form of anonymous authorization like a start button. The end of authorization will cause the Charging Station to stop the energy transfer, after which the TransactionEventRequest with eventType = Ended will be transmitted.
PowerPathClosed	All preconditions for charging are no longer met. This event is the logical OR of EVConnected and Authorized and should be used if a transaction is supposed to end when EV is disconnected and/or deauthorized. This will cause the Charging Station to stop the energy transfer, after which the TransactionEventRequest with eventType = Ended will be transmitted. It is exactly the same as having the values EVConnected, Authorized in TxStopPoint . Despite its name, this event is not related to the state of the power relay.

19. Appendix 1

19.1. Page 2 - (v1) InvalidFirmwareSignature/SigningCertificate are critical security events [682]

The column "Critical" must be set to "yes" for a security event InvalidFirmwareSignature and InvalidFirmwareSigningCertificate, because of the SHALL-requirements L01.FR.02 and L01.FR.03.

Security Event	Description	Critical
InvalidFirmwareSignature	The firmware signature is not valid	Yes
InvalidFirmwareSigningCertificate	The certificate used to verify the firmware signature is not valid	Yes

20. Appendix 3

20.1. Page 9 - (v1) OCPPCommCtrlr.ActiveNetworkProfile must be of type integer [697]

ActiveNetworkProfile was mistakenly shown as having type string. This must be integer.

OCPPCommCtrlr

Description		
Logical Component responsible for configuration relating to information exchange between Charging Station and CSMS.		
Variables	Type	Description
ActiveNetworkProfile	integer	[...]

20.2. Page 10 - (v1) SecurityCtrlr.BasicAuthPassword and Identity should have dataType=string [698]

BasicAuthPassword was shown as type "passwordString" and for Identity as type "identifierString". The type for the device model variable in both cases must be "string".

SecurityCtrlr

Description		
Logical Component responsible for configuration relating to security of communications between Charging Station and CSMS.		
Variables	Type	Description
BasicAuthPassword	string	[...]
Identity	string	[...]



OCPP 2.0.1

Part 2 - Specification

Edition 2 FINAL, 2022-12-15

Table of Contents

Disclaimer	1
Generic	2
Version History	3
1. Scope	4
1.1. OCPP 2.0.1	4
1.2. OCPP 2.0.1 Edition 2	4
2. Conventions, Terminology and Abbreviations	5
2.1. Conventions	5
2.2. Terminology	6
2.3. Abbreviations	8
2.4. Actors	10
2.5. References	11
2.6. Definition of Transaction	12
2.7. ISO 15118 support	14
3. Generic Requirements	15
3.1. Time Format Requirements	15
3.2. Message Timeouts	16
3.3. Language support	16
A. Security	17
1. OCPP Security	18
1.1. Security Objectives	18
1.2. Design Considerations	18
1.3. Security Profiles	19
1.4. Keys used in OCPP	26
1.5. Certificate Revocation	28
1.6. Installation	29
2. Use cases & Requirements	30
A01 - Update Charging Station Password for HTTP Basic Authentication	30
A02 - Update Charging Station Certificate by request of CSMS	31
A03 - Update Charging Station Certificate initiated by the Charging Station	35
A04 - Security Event Notification	39
A05 - Upgrade Charging Station Security Profile	40
B. Provisioning	42
1. Introduction	43
1.1. Transactions before being accepted by a CSMS	43
2. Use cases & Requirements	44
2.1. Booting a Charging Station	44
B01 - Cold Boot Charging Station	44
B02 - Cold Boot Charging Station - Pending	47
B03 - Cold Boot Charging Station - Rejected	50
B04 - Offline Behavior Idle Charging Station	52
2.2. Configuring a Charging Station	53
B05 - Set Variables	53
B06 - Get Variables	55
B07 - Get Base Report	57
B08 - Get Custom Report	60
B09 - Setting a new NetworkConnectionProfile	62
B10 - Migrate to new CSMS	63
2.3. Resetting a Charging Station	64
B11 - Reset - Without Ongoing Transaction	64
B12 - Reset - With Ongoing Transaction	67
C. Authorization	70
1. Introduction	71
1.1. ID Tokens	71
1.2. Group ID Tokens	71
1.3. Authorization Cache	72

1.4. Local Authorization List	72
1.5. Unknown Offline Authorization	72
2. Use cases & Requirements.....	73
2.1. Authorization options	73
C01 - EV Driver Authorization using RFID.....	73
C02 - Authorization using a start button.....	77
C03 - Authorization using credit/debit card.....	79
C04 - Authorization using PIN-code	82
C05 - Authorization for CSMS initiated transactions	84
C06 - Authorization using local id type.....	86
2.2. ISO 15118 Authorization	89
C07 - Authorization using Contract Certificates	89
C08 - Authorization at EVSE using ISO 15118 External Identification Means (EIM).....	92
2.3. GroupId	94
C09 - Authorization by GroupId	94
2.4. Authorization Cache	96
C10 - Store Authorization Data in the Authorization Cache.....	96
C11 - Clear Authorization Data in Authorization Cache	98
C12 - Start Transaction - Cached Id	99
2.5. Local Authorization list	101
C13 - Offline Authorization through Local Authorization List	101
C14 - Online Authorization through Local Authorization List.....	102
2.6. Offline Authorization	104
C15 - Offline Authorization of unknown Id	104
2.7. Master Pass	106
C16 - Stop Transaction with a Master Pass.....	106
D. LocalAuthorizationList Management	109
1. Introduction	110
2. Use cases & Requirements	111
D01 - Send Local Authorization List	111
D02 - Get Local List Version.....	114
E. Transactions.....	115
1. Introduction	116
1.1. Flexible transaction start/stop.....	116
1.2. TransactionId generation	117
1.3. Delivering transaction-related messages.....	117
1.4. Authorization	118
1.5. Clarification for optional fields in TransactionEventRequest.....	118
2. Use cases & Requirements	120
2.1. OCPP transaction mechanism.....	120
E01 - Start Transaction options	120
E02 - Start Transaction - Cable Plugin First	126
E03 - Start Transaction - IdToken First	131
E04 - Transaction started while Charging Station is offline.....	135
E05 - Start Transaction - Id not Accepted	139
E06 - Stop Transaction options	142
E07 - Transaction locally stopped by IdToken	147
E08 - Transaction stopped while Charging Station is offline.....	151
E09 - When cable disconnected on EV-side: Stop Transaction	154
E10 - When cable disconnected on EV-side: Suspend Transaction	157
E11 - Connection Loss During Transaction	160
E12 - Inform CSMS of an Offline Occurred Transaction.....	162
E13 - Transaction-related message not accepted by CSMS	164
E14 - Check transaction status	166
2.2. Interrupting and Stopping ISO 15118 Charging	167
E15 - End of charging process.....	167
F. RemoteControl	169
1. Introduction	170
2. Use cases & Requirements	171

2.1. Remote Transaction Control	171
F01 - Remote Start Transaction - Cable Plugin First.....	171
F02 - Remote Start Transaction - Remote Start First	175
F03 - Remote Stop Transaction.....	180
F04 - Remote Stop ISO 15118 Charging from CSMS	182
2.2. Unlock Connector	184
F05 - Remotely Unlock Connector.....	184
2.3. Remote Trigger	186
F06 - Trigger Message	186
G. Availability	189
1. Introduction	190
2. Use cases & Requirements	191
G01 - Status Notification	191
G02 - Heartbeat	193
G03 - Change Availability EVSE/Connector	195
G04 - Change Availability Charging Station	197
G05 - Lock Failure	199
H. Reservation	201
1. Introduction	202
2. Use cases & Requirements	203
H01 - Reservation	203
H02 - Cancel Reservation.....	207
H03 - Use a reserved EVSE	208
H04 - Reservation Ended, not used.....	211
I. TariffAndCost	212
1. Introduction	213
1.1. Why no structured tariff information?.....	213
2. Use cases & Requirements	214
I01 - Show EV Driver-specific Tariff Information.....	214
I02 - Show EV Driver Running Total Cost During Charging.....	215
I03 - Show EV Driver Final Total Cost After Charging.....	216
I04 - Show Fallback Tariff Information	217
I05 - Show Fallback Total Cost Message	219
I06 - Update Tariff Information During Transaction	220
J. MeterValues	222
1. Introduction	223
2. Configuration	224
2.1. Transaction Meter Values	224
2.2. Clock-Aligned Meter Values	224
2.3. Multiple Locations/Phases.....	225
2.4. Signed Meter Values	225
3. Use cases & Requirements	226
3.1. MeterValues	226
J01 - Sending Meter Values not related to a transaction	226
J02 - Sending transaction related Meter Values	229
3.2. ISO 15118 MeterValue signing	231
J03 - Charging Loop with metering information exchange	231
K. SmartCharging	233
1. Introduction	234
2. Types of Smart Charging	235
2.1. Internal Load Balancing	235
2.2. Central Smart Charging.....	235
2.3. Local Smart Charging	236
2.4. External Smart Charging Control Signals	236
3. Charging profiles.....	238
3.1. Introduction	238
3.2. Charging profile purposes	238
3.3. Charging profile recurrency	238
3.4. Stacking charging profiles	239

3.5. Combining Charging Profile Purposes	239
3.6. Example Charging Profile	240
4. Smart Charging Signals to a Charging Station from Multiple Actors	242
5. Use cases & Requirements	243
5.1. General Smart Charging	243
K01 - SetChargingProfile	243
K02 - Central Smart Charging	248
K03 - Local Smart Charging	252
K04 - Internal Load Balancing	255
K05 - Remote Start Transaction with Charging Profile	256
K06 - Offline Behavior Smart Charging During Transaction	258
K07 - Offline Behavior Smart Charging at Start of Transaction	259
K08 - Get Composite Schedule	261
K09 - Get Charging Profiles	263
K10 - Clear Charging Profile	264
5.2. External Charging Limit based Smart Charging	267
K11 - Set / Update External Charging Limit With Ongoing Transaction	267
K12 - Set / Update External Charging Limit Without Ongoing Transaction	269
K13 - Reset / Release External Charging Limit	270
K14 - External Charging Limit with Local Controller	272
5.3. ISO 15118 based Smart Charging	274
K15 - Charging with load leveling based on High Level Communication	274
K16 - Renegotiation initiated by CSMS	277
K17 - Renegotiation initiated by EV	279
L. FirmwareManagement	283
1. Introduction	284
2. Use cases & Requirements	285
L01 - Secure Firmware Update	285
L02 - Non-Secure Firmware Update	290
L03 - Publish Firmware file on Local Controller	293
L04 - Unpublish Firmware file on Local Controller	295
M. ISO 15118 CertificateManagement	297
1. Introduction	298
2. ISO 15118 Certificates	301
2.1. ISO 15118 Certificate structure	301
2.2. Using ISO 15118 Certificates in OCPP	302
2.3. 15118 communication set-up	303
2.4. Certificate - Use Case mapping	303
3. Use cases from ISO 15118 relevant for OCPP	305
4. Use cases & Requirements	306
M01 - Certificate installation EV	306
M02 - Certificate Update EV	307
M03 - Retrieve list of available certificates from a Charging Station	308
M04 - Delete a specific certificate from a Charging Station	309
M05 - Install CA certificate in a Charging Station	310
M06 - Get V2G Charging Station Certificate status	312
N. Diagnostics	314
1. Introduction	315
2. Use cases & Requirements	316
2.1. Logging	316
N01 - Retrieve Log Information	316
2.2. Configure Monitoring	318
N02 - Get Monitoring report	318
N03 - Set Monitoring Base	320
N04 - Set Variable Monitoring	321
N05 - Set Monitoring Level	324
N06 - Clear / Remove Monitoring	325
2.3. Monitoring Events	326
N07 - Alert Event	326

N08 - Periodic Event	328
2.4. Customer Information	330
N09 - Get Customer Information	330
N10 - Clear Customer Information	331
O. DisplayMessage	334
1. Introduction	335
2. Use cases & Requirements	336
001 - Set DisplayMessage	336
002 - Set DisplayMessage for Transaction	338
003 - Get All DisplayMessages	340
004 - Get Specific DisplayMessages	342
005 - Clear a DisplayMessage	344
006 - Replace DisplayMessage	345
P. DataTransfer	346
1. Introduction	347
2. Use cases & Requirements	348
P01 - Data Transfer to the Charging Station	348
P02 - Data Transfer to the CSMS	350
Messages, Datatypes & Enumerations	352
1. Messages	353
1.1. Authorize	353
1.2. BootNotification	353
1.3. CancelReservation	354
1.4. CertificateSigned	354
1.5. ChangeAvailability	355
1.6. ClearCache	355
1.7. ClearChargingProfile	356
1.8. ClearDisplayMessage	356
1.9. ClearedChargingLimit	356
1.10. ClearVariableMonitoring	357
1.11. CostUpdated	357
1.12. CustomerInformation	358
1.13. DataTransfer	358
1.14. DeleteCertificate	359
1.15. FirmwareStatusNotification	359
1.16. Get15118EVCertificate	360
1.17. GetBaseReport	360
1.18. GetCertificateStatus	361
1.19. GetChargingProfiles	361
1.20. GetCompositeSchedule	362
1.21. GetDisplayMessages	362
1.22. GetInstalledCertificateIds	363
1.23. GetLocalListVersion	363
1.24. GetLog	363
1.25. GetMonitoringReport	364
1.26. GetReport	365
1.27. GetTransactionStatus	365
1.28. GetVariables	366
1.29. Heartbeat	366
1.30. InstallCertificate	366
1.31. LogStatusNotification	367
1.32. MeterValues	367
1.33. NotifyChargingLimit	367
1.34. NotifyCustomerInformation	368
1.35. NotifyDisplayMessages	368
1.36. NotifyEVChargingNeeds	369
1.37. NotifyEVChargingSchedule	369
1.38. NotifyEvent	370
1.39. NotifyMonitoringReport	370

1.40. NotifyReport	371
1.41. PublishFirmware	371
1.42. PublishFirmwareStatusNotification	372
1.43. ReportChargingProfiles	372
1.44. RequestStartTransaction	373
1.45. RequestStopTransaction	373
1.46. ReservationStatusUpdate	374
1.47. ReserveNow	374
1.48. Reset	375
1.49. SecurityEventNotification	375
1.50. SendLocalList	375
1.51. SetChargingProfile	376
1.52. SetDisplayMessage	377
1.53. SetMonitoringBase	377
1.54. SetMonitoringLevel	378
1.55. SetNetworkProfile	379
1.56. SetVariableMonitoring	379
1.57. SetVariables	379
1.58. SignCertificate	380
1.59. StatusNotification	380
1.60. TransactionEvent	381
1.61. TriggerMessage	382
1.62. UnlockConnector	383
1.63. UnpublishFirmware	383
1.64. UpdateFirmware	384
2. Datatypes	385
2.1. ACChargingParametersType	385
2.2. AdditionalInfoType	385
2.3. APNType	385
2.4. AuthorizationData	386
2.5. CertificateHashDataChainType	386
2.6. CertificateHashDataType	386
2.7. ChargingLimitType	386
2.8. ChargingNeedsType	387
2.9. ChargingProfileCriterionType	387
2.10. ChargingProfileType	387
2.11. ChargingSchedulePeriodType	388
2.12. ChargingScheduleType	388
2.13. ChargingStationType	389
2.14. ClearChargingProfileType	389
2.15. ClearMonitoringResultType	390
2.16. ComponentType	390
2.17. ComponentVariableType	390
2.18. CompositeScheduleType	390
2.19. ConsumptionCostType	391
2.20. CostType	391
2.21. DCChargingParametersType	391
2.22. EventDataType	392
2.23. EVSETType	392
2.24. FirmwareType	393
2.25. GetVariableDataType	393
2.26. GetVariableResultType	393
2.27. IdTokenInfoType	394
2.28. IdTokenType	394
2.29. LogParametersType	395
2.30. MessageContentType	395
2.31. MessageInfoType	395
2.32. MeterValueType	396
2.33. ModemType	396

2.34. MonitoringDataType	396
2.35. NetworkConnectionProfileType	396
2.36. OCSPRequestDataType	397
2.37. RelativeTimeIntervalType	397
2.38. ReportDataType	397
2.39. SalesTariffEntryType	398
2.40. SalesTariffType	398
2.41. SampledValueType	398
2.42. SetMonitoringDataType	399
2.43. SetMonitoringResultType	400
2.44. SetVariableDataType	401
2.45. SetVariableResultType	402
2.46. SignedMeterValueType	402
2.47. StatusInfoType	402
2.48. TransactionType	403
2.49. UnitOfMeasureType	403
2.50. VariableAttributeType	403
2.51. VariableCharacteristicsType	404
2.52. VariableMonitoringType	404
2.53. VariableType	405
2.54. VPNTYPE	406
3. Enumerations	407
3.1. APNAuthenticationEnumType	407
3.2. AttributeEnumType	407
3.3. AuthorizationStatusEnumType	407
3.4. AuthorizeCertificateStatusEnumType	407
3.5. BootReasonEnumType	408
3.6. CancelReservationStatusEnumType	408
3.7. CertificateActionEnumType	408
3.8. CertificateSignedStatusEnumType	409
3.9. CertificateSigningUseEnumType	409
3.10. ChangeAvailabilityStatusEnumType	409
3.11. ChargingLimitSourceEnumType	409
3.12. ChargingProfileKindEnumType	410
3.13. ChargingProfilePurposeEnumType	410
3.14. ChargingProfileStatusEnumType	410
3.15. ChargingRateUnitEnumType	410
3.16. ChargingStateEnumType	411
3.17. ClearCacheStatusEnumType	411
3.18. ClearChargingProfileStatusEnumType	411
3.19. ClearMessageStatusEnumType	412
3.20. ClearMonitoringStatusEnumType	412
3.21. ComponentCriterionEnumType	412
3.22. ConnectorEnumType	412
3.23. ConnectorStatusEnumType	413
3.24. CostKindEnumType	413
3.25. CustomerInformationStatusEnumType	414
3.26. DataEnumType	414
3.27. DataTransferStatusEnumType	414
3.28. DeleteCertificateStatusEnumType	414
3.29. DisplayMessageStatusEnumType	415
3.30. EnergyTransferModeEnumType	415
3.31. EventNotificationEnumType	415
3.32. EventTriggerEnumType	415
3.33. FirmwareStatusEnumType	416
3.34. GenericDeviceModelStatusEnumType	416
3.35. GenericStatusEnumType	417
3.36. GetCertificateIdUseEnumType	417
3.37. GetCertificateStatusEnumType	417

3.38. GetChargingProfileStatusEnumType	417
3.39. GetDisplayMessagesStatusEnumType.....	417
3.40. GetInstalledCertificateStatusEnumType	418
3.41. GetVariableStatusEnumType	418
3.42. HashAlgorithmEnumType	418
3.43. IdTokenEnumType	418
3.44. InstallCertificateStatusEnumType.....	419
3.45. InstallCertificateUseEnumType	419
3.46. Iso15118EVCertificateStatusEnumType	419
3.47. LocationEnumType	419
3.48. LogEnumType	420
3.49. LogStatusEnumType.....	420
3.50. MeasurandEnumType	420
3.51. MessageFormatEnumType	421
3.52. MessagePriorityEnumType	422
3.53. MessageStateEnumType	422
3.54. MessageTriggerEnumType	422
3.55. MonitorEnumType	423
3.56. MonitoringBaseEnumType.....	423
3.57. MonitoringCriterionEnumType.....	423
3.58. MutabilityEnumType	423
3.59. NotifyEVChargingNeedsStatusEnumType.....	424
3.60. OCPPInterfaceEnumType.....	424
3.61. OCPPTransportEnumType	424
3.62. OCPPVersionEnumType	424
3.63. OperationalStatusEnumType	425
3.64. PhaseEnumType	425
3.65. PublishFirmwareStatusEnumType	425
3.66. ReadingContextEnumType	426
3.67. ReasonEnumType	426
3.68. RecurrencyKindEnumType	427
3.69. RegistrationStatusEnumType	427
3.70. ReportBaseEnumType	427
3.71. RequestStartStopStatusEnumType	428
3.72. ReservationUpdateStatusEnumType	428
3.73. ReserveNowStatusEnumType	428
3.74. ResetEnumType	429
3.75. ResetStatusEnumType	429
3.76. SendLocalListStatusEnumType	429
3.77. SetMonitoringStatusEnumType	429
3.78. SetNetworkProfileStatusEnumType	430
3.79. SetVariableStatusEnumType	430
3.80. TransactionEventEnumType	430
3.81. TriggerMessageStatusEnumType	430
3.82. TriggerReasonEnumType	431
3.83. UnlockStatusEnumType	431
3.84. UnpublishFirmwareStatusEnumType	432
3.85. UpdateEnumType	432
3.86. UpdateFirmwareStatusEnumType	432
3.87. UploadLogStatusEnumType	432
3.88. VPNEnumType.....	433
Referenced Components and Variables	434
1. Controller Components	435
2. Referenced Components and Variables	436
2.1. General	436
2.2. Security related	444
2.3. Authorization related.....	446
2.4. Authorization Cache related.....	448
2.5. Local Authorization List Management related.....	449

2.6. Transaction related	450
2.7. Metering related.	453
2.8. Reservation related	458
2.9. Smart Charging related.	458
2.10. Tariff & Cost related	461
2.11. Diagnostics related	462
2.12. Display Message related	464
2.13. Charging Infrastructure related	465
2.14. ISO 15118 Related.	468

Disclaimer

Copyright © 2010 – 2022 Open Charge Alliance. All rights reserved.

This document is made available under the **Creative Commons Attribution-NoDerivatives 4.0 International Public License** (<https://creativecommons.org/licenses/by-nd/4.0/legalcode>).

Generic

Version History

Version	Date	Description
2.0.1 Edition 2	2022-12-15	OCPP 2.0.1 Edition 2. All errata from OCPP 2.0.1 Part 2 - Errata v2.0 have been merged into this version of the specification.
2.0.1	2020-03-31	Final version of OCPP 2.0.1
2.0	2018-04-11	OCPP 2.0 April 2018 First major release since 1.0. Lots of new/improved/revised functionality Revised documentation

1. Scope

This document defines the protocol used between a **Charging Station** and a **Charging Station Management System** in an EV charging infrastructure in the form of use cases. If the protocol requires a certain action or response from one side or the other, then this will be stated in this document.

This part of the specification does not define the communication technology. In order to ensure widespread compatibility OCPP 2.0.1 is limited to JSON. The specifications for the JSON implementation are in "Part 4 - JSON over WebSockets implementation guide".

1.1. OCPP 2.0.1

This specification defines version 2.0.1 of OCPP.

After the release of OCPP 2.0, some issues were found in OCPP 2.0. Some of these issues could not be fixed issuing errata to the specification text only, as has been done with OCPP 1.6, but required changes to the protocol's machine-readable schema definition files that cannot be backward compatible.

To prevent confusion in the market and possible interoperability issues in the field, OCA has decided to name this version: 2.0.1. OCPP 2.0.1 contains fixes for all the known issues, to date, not only the fixes to the messages.

This version replaces OCPP 2.0. OCA advises implementers of OCPP to no longer implement OCPP 2.0 and only use version 2.0.1 going forward.

As a rule, existing numbered requirements are only updated or removed, previously used requirements numbers are never reused for a totally different requirement.

Any mentions of "OCPP 2.0" refers to revision 2.0.1 unless specifically stated otherwise.

1.2. OCPP 2.0.1 Edition 2

Two errata have been released for part 2 of the OCPP 2.0.1 specification.: "OCPP-2.0.1_part2_errata_v1_0" was released in 2021. In 2022 this was extended with additional errata entries in "OCPP-2.0.1_part2_errata_v2_0".

These errata have been incorporated in this document, "OCPP-2.0.1_part2_specification_edition2", such that it is no longer necessary to read the errata in addition to the specification. The incorporation of the errata in edition 2 does not affect any schemas of OCPP messages. Certain errata did contain changes to requirements or even new requirements, but only in cases where a requirement contains an obvious error and would not or could not be implemented literally. New requirements were only added when they were already implicitly there. These changes have been discussed in or were proposed by the Technology Working Group of the Open Charge Alliance.

The appendices of the OCPP 2.0.1 part 2 can be updated without requiring a new OCPP release. This mainly concerns the components and variables of the OCPP device model, which can be extended with new components or variables, as long as they are optional.

2. Conventions, Terminology and Abbreviations

2.1. Conventions

2.1.1. Normative

All sections and appendices are normative, unless they are explicitly indicated to be informative.

2.1.2. Requirements take precedence over text

Whenever there is any (apparent) conflict between narrative text and requirements in the specification document, the requirements have precedence.

2.1.3. Requirement Keywords

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC-2119 [[RFC2119](#)], subject to the following additional clarification clause:

The phrase "valid reasons in particular circumstances" relating to the usage of the terms "SHOULD", "SHOULD NOT", "RECOMMENDED", and "NOT RECOMMENDED" is to be taken to mean technically valid reasons, such as the absence of necessary hardware to support a function from a Charging Station design: for the purposes of this specification it specifically excludes decisions made on commercial, or other non-technical grounds, such as cost of implementation, or likelihood of use.

2.1.4. Primitive Datatypes

The specification mentions the following primitive datatypes:

Table 1. Primitive Datatypes

Datatype	Description
string	The characters defined in the UTF-8 character set are allowed to be used.
integer	32 bit (31 bit resolution, 1 sign bit) No leading 0's No plus sign Allowed value examples: 1234, -1234 Not Allowed: 01234, +1234
decimal	For data being reported by the Charging Station, the full resolution of the source data must be preserved. The decimal sent towards the Charging Station SHALL NOT have more than six decimal places.
identifierString	This is a case-insensitive dataType and can only contain characters from the following character set: a-z, A-Z, 0-9, '*', '^', '_', '=', ' ', '+', '!', '@', ''
dateTime	All time values exchanged between CSMS and Charging Station SHALL be formatted as defined in [RFC3339]. Additionally fractional seconds have been given an extra limit. The number of decimal places SHALL NOT exceed the maximum of 3. Example 1: 2019-04-12T23:20:50.52Z represents 20 minutes and 50.52 seconds after the 23rd hour of April 12th, 2019 in UTC. Example 2: 2019-12-19T16:39:57+01:00 represents 39 minutes and 57 seconds after the 16th hour of December 19th, 2019 with an offset of +01:00 from UTC (Central European Time).
passwordString	This is a UTF-8 encoded case-sensitive string that can only contain characters from the following character set: "a-z", "A-Z", "0-9" or any of the following limited set of symbols: * - _ = : + @ .
AnyType	Text, data without specified length or format.
boolean	Only allowed values: "false" and "true".

2.1.5. Normal communication

Unless otherwise specified, all use cases and requirements assume normal communication between Charging Station and CSMS (*Online*).

2.1.6. Field description

In many cases, further explanation about how or when to use certain fields in messages and datatypes is given in the field description. See Chapter [Messages](#).

2.2. Terminology

2.2.1. General Terminology

This section contains the terminology that is used throughout this document.

Table 2. Terminology

Terminology	Description
Application layer	OSI-Layer 5-7.
Authentication	Authentication is the process of confirming an identity or attribute. When speaking about authentication one should distinguish between user authentication (e.g. sender/receiver) and message authentication.
Block cipher	Cryptographic primitive to encrypt/decrypt messages of fixed block length. Example: AES encrypts blocks of 128 bits (16 bytes) at a time.
Cable Plugged in	In this document this can mean the following: - Cable fixed on Charging Station side, cable plugged in to EV - Cable plugged into the Charging Station and EV - Wireless Charger detects an EV
Certificate	A digital certificate authenticates a public key or entity. See also Public-Key Infrastructure.
Certificate Management Protocol	An internet protocol used to manage X.509 digital certificates within a PKI. It is described in RFC 4210 and uses the certificate request message format (CRMF) described in RFC 4211.
Charging Cable	Cable assembly equipped with a, by the EV accepted, plug, intended to be used for the connection between an EV and an EVSE. One side may be permanently attached to the EVSE, or also be equipped with a plug that is accepted by the EVSE.
Charging Loop	In this specification the ISO 15118-2 definition of the charging loop is used: <i>the V2G messaging phase for controlling the charging process by ISO 15118</i> .
Charging Profile	Generic Charging Profile, used for different types of Profiles. Contains information about the Profile and holds the ChargingSchedule .
Charging Schedule	Part of a Charging Profile. Defines a block of charging Power or Current limits. Can contain a start time and length.
Charging Station	The Charging Station is the physical system where EVs can be charged. A Charging Station has one or more EVSEs.
Composite Charging Schedule	The charging schedule as calculated by the Charging Station. It is the result of the calculation of all active schedules and possible local limits present in the Charging Station. Local Limits might be taken into account.
Confidentiality	Only authorized entities may access confidential data. To protect data from unauthorized access it can be encrypted. Then only entities with access to the secret keys can access the data after decrypting it.
Connector	The term Connector, as used in this specification, refers to an independently operated and managed electrical outlet on a Charging Station. In other words, this corresponds to a single physical Connector. In some cases an EVSE may have multiple physical socket types and/or tethered cable/Connector arrangements(i.e. Connectors) to facilitate different vehicle types (e.g. four-wheeled EVs and electric scooters).
Contactor	An electrically controlled switching device, typically used by Charging Stations to switch charging power on/off.
Contract Certificate	A valid certificate for a charging contract in an EV for 15118 communication.
Control Pilot signal	A signal used by a Charging Station to inform an EV of a maximum current limit, as defined by IEC61851-1 .

Terminology	Description
Cost	Cost to be paid by an EV Driver for consumed energy/time etc. Including taxes.
Cryptographic hash function	Cryptographic hash functions should behave as one-way functions. They must be preimage resistant, 2nd preimage resistant, and collision-resistant. Changes in the input must produce explicitly different results in the output. Example: SHA-256. See also ENISA OCPP Security [1] .
Cryptography	The ENISA Algorithms, Key Sizes and Parameters Report [1] provides an overview of the current state of the art.
CSMS	Charging Station Management System. The system that manages Charging Stations and has the information for authorizing Users for using its Charging Stations.
Data Integrity	See Integrity and Message authentication.
Digital Signature	Authenticates the sender. In practice digital signatures are implemented using elliptic curves (EC).
Encryption	Using a cryptographic scheme, the message is mapped to a random-looking undecipherable string (ciphertext). Decryption reverses the encryption process and can only be performed with the corresponding decryption key. This decryption key is either the same as the encryption key (symmetric cryptography) or the private key in a public-key cryptosystem. The confidentiality of the message can be guaranteed only while the keys are kept secret.
Energy Management System	A device that manages the local loads (consumption and production) based on local and/or contractual constraints and/or contractual incentives. It has additional inputs, such as sensors and controls from e.g. PV, battery storage.
Energy Offer Period	Time during which a Charging Station is ready and willing to offer energy to an EV.
Energy Transfer Period	Time during which an EV chooses to take offered energy, or return it.
EVSE	An EVSE is considered as an independently operated and managed part of the Charging Station that can deliver energy to one EV at a time.
Hash function	Function that maps a message to a bit string of fixed length (hash value). See also cryptographic hash function.
Hash value	Output of a (cryptographic) hash function. The length is fixed in the specs of the hash function.
High level communication	bi-directional digital communication using protocol and messages and physical and data link layers specified in ISO 15118 series [ISO15118-1]
Idle State	In both use cases and sequence diagrams, <i>Idle</i> status is referred as the state in which a Charging Station is not performing any use case related tasks. Condition during which the equipment can promptly provide a primary function but is not doing so.
Integrity	Data cannot be altered without authorization. See also Message authentication.
Local Controller	A logical entity between a CSMS and one or more Charging Stations that has the ability to control charging of a group of Charging Stations based on the input from the CSMS, and can send messages to its Charging Stations, independently of the CSMS.
Master Pass	IdToken that can be used to stop any (or all) ongoing transactions. This can be used by for example law enforcement personal to stop a transaction.
Master Pass UI	Master Pass User Interface, this might be a full color touchscreen, but might also be just a couple of buttons and LEDs and/or sounds that enable a user to select transactions to be stopped.
Message authentication	Messages should be protected against unauthorized modifications. The message should always be sent together with an authentication tag providing its authenticity. Such an authentication tag can be the second output of an authenticated cipher such as AES-CCM or AES-GCM or a message authentication code.
Mode of Operation	A mode of operation specifies how the message blocks are processed by the block cipher. Using a block cipher in CBC or CTR mode provides encryption only, whereas using a block cipher in CCM or GCM mode encrypts the plaintext and produces a message authentication tag for the ciphertext.
OCPP-J	OCPP via JSON over WebSocket.
Offline	There is no communication possible between the Charging Station and CSMS. For an OCPP-J connection this means the WebSocket connection is not open.
Password authentication	The user proves his/her identity using a password or PIN.
Phase Rotation	Defines the wiring order of the phases between the electrical meter (or if absent, the grid connection), and the Charging Station Connector.
Price	Specific price tag of a single tariff entry, for example: 0.35 per kWh incl. 18% VAT.

Terminology	Description
Public-key cryptography	"Cryptographic scheme where a public key is published and henceforth can be used for encryption of messages or verification of digital signatures. Each public key has a counterpart, the corresponding private key. This key must be kept secret and is used for decryption or digital signing of messages. Public-key primitives have a high computational complexity for encryption and therefore are mostly used as part of a hybrid encryption scheme where the public key is used to communicate a common symmetric session key under which all further communication is encrypted. Certificates administered by a public-key infrastructure are used to establish the authenticity of the public key. See also ENISA OCPP Security [12]. The most popular public-key encryption scheme is RSA. Digital signatures can be generated most efficiently with elliptic-curve based (EC) mechanisms."
Public-key infrastructure	System to generate, administer, and revoke certificates.
Resume regular transaction	Used in sequence diagrams to indicate that this use case/sequence diagram has ended, but the transaction has not ended and will continue, but that is outside of scope of that specific use case.
Requirement	Provision that conveys criteria to be fulfilled. ISO/IEC Guide 2:2004, 7.5.
Security Event	Any event relevant to the secure operation of the device.
Security Function	Any function on the device that is needed for it to be operated securely, including access control, authentication, and encryption.
Session	A Session in OCPP is a general term that refers to the charging process of an EV, that might include a Transaction.
Session key	Symmetric key with a limited lifetime.
Symmetric cryptography	Sender and receiver hold the same key. Examples for symmetric primitives are block ciphers or MACs.
Transaction	A transaction in OCPP is a part of the complete process of charging an EV that starts and stops based on configurable parameters. These configurable parameters refer to moments in the charging process, such as the EV being connected or the EV driver being authorized.
Tariff	Collection of prices depending on charging time, power usage and other price affecting parameters.
Use case	A use case is a structured way of describing the (inter)actions necessary to achieve a certain objective. In this document, a use case consists of an actor list, a scenario description, postconditions and a sequence diagram and is always followed by a list of numbered requirements.
User Authentication	Verification of the identity of the communication partners (e.g., user on the device). Moreover, verification that the communication partners are still alive throughout a session.

2.2.2. ISO 15118 and OCPP terminology mapping

This section is informative.

The ISO 15118 terminology is more comprehensive when referring to specific components within EVs and Charging Stations. The following table shows a "mapping" of these terms.

Table 3. ISO 15118 and OCPP terminology mapping

ISO 15118	OCPP
ChargingProfile (contains the power over time the EV is planned to consume)	Loosely corresponds to ChargingSchedule in NotifyEVChargingSchedule message.
SASchedule (the power limits from a secondary actor for charging an EV for a specific time)	Loosely corresponds to ChargingProfile in SetChargingProfile message.
EVCC (i.e. Electric Vehicle Communication Controller)	Controller in the EV that is used for ISO 15118 communication.
Outlet	Connector
SECC (i.e. Supply Equipment Communication Controller)	Controller in the EVSE of the Charging Station that is used for ISO 15118 communication.
SA (i.e. Secondary Actor)	CSMS (or other backend systems)

2.3. Abbreviations

2.3.1. General Abbreviations

This section contains the abbreviations that are used throughout this document.

Table 4. Abbreviations

Abbreviation	Description
AES	Advanced Encryption Standard. Original name for this block cipher was Rijndael named after its designers Vincent Rijmen and Joan Daemen.
BEV	Battery Electric Vehicle
CMP	Certificate Management Protocol
CS	Charging Station
CSL	Comma Separated List
CSMS	Charging Station Management System
CSO	Charging Station Operator
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
DSO	Distribution System Operator
DST	Daylight Saving Time
EC	Elliptic Curve. See also ENISA OCPP Security [1]
ECDSA	Elliptic Curve Digital Signature Algorithm.
EMS	Energy Management System
ENISA	European Union Agency for Network and Information Security.
EV	Electric Vehicle
EVSE	EV Supply Equipment IEC61851-1
FQDN	Fully Qualified Domain Name
FTP(S)	File Transport Protocol (Secure)
HTTP(S)	HyperText Transport Protocol (Secure)
ICCID	Integrated Circuit Card Identifier
IMSI	International Mobile Subscription Identity
JSON	JavaScript Simple Object Notation
MAC	Message authentication code. Provides data integrity. Examples: CMAC, GMAC. See also ENISA OCPP Security [1] .
NAT	Network Address Translation
NIST	National Institute of Standards and Technology.
NTP	Network Time Protocol
PDU	Protocol Data Unit
PHEV	Plugin Hybrid Electric Vehicle
RDN	Relative Distinguished Name
RSA	Public-key cryptosystem named after its inventors Rivest, Shamir, and Adleman.
RSA-PSS	RSA-PSS is a new signature scheme that is based on the RSA cryptosystem and provides increased security assurance. It was added in version 2.1 of PKCS #1, following OCPP Security [23]
RST	3 phase power connection, Standard Reference Phasing
RTS	3 phase power connection, Reversed Reference Phasing
SRT	3 phase power connection, Reversed 240 degree rotation
STR	3 phase power connection, Standard 120 degree rotation
TRS	3 phase power connection, Standard 240 degree rotation
TSR	3 phase power connection, Reversed 120 degree rotation
SC	Smart Charging
TLS	Transport Layer Security
TSO	Transmission System Operator
URI	Uniform Resource Identifier RFC-3986 [RFC3986]
URL	Uniform Resource Locator - refers to the subset of URLs that, in addition to identifying a resource, provide a means of locating the resource by describing its primary access mechanism (e.g., its network "location").

Abbreviation	Description
UTC	Coordinated Universal Time
WAN	Wide Area Network.

2.3.2. ISO 15118 Abbreviations

This section contains the abbreviations from ISO 15118 that are used in this document.

Table 5. ISO 15118 Abbreviations

EIM	External Identification Means
EMAID	E-Mobility Account Identifier
EVCC	EV Communication Controller
HLC	High Level Communication
HMI	Human Machine Interface
LAN	Local Area Network
MO	Mobility Operator
OEM	Original Equipment Manufacturer
OCSP	Online Certificate Status Protocol
PWM	Pulse Width Modulation
SA	Secondary Actor
SECC	Supply Equipment Communication Controller
V2G	Vehicle to Grid

2.4. Actors

This section is informative.

In OCPP, system actors are covering functions or devices.

Table 6. Actors

Actor name	Actor type	Actor description
EV Driver	Actor	The Driver of an EV who wants to charge the EV at a Charging Station.
Connector	Device	The term "Connector", as used in this specification, refers to an independently operated and managed electrical outlet on a Charging Station. In other words, this corresponds to a single physical Connector. In some cases an EVSE may have multiple Connectors: multiple physical socket types and/or types (e.g. four-wheeled EVs and electric scooters).
CSMS	System	Charging Station Management System: manages Charging Stations and has the information for authorizing Users for using its Charging Stations.
Charging Station	Device	The Charging Station is the physical system where an EV can be charged. A Charging Station has one or more EVSEs.
Charging Station Operator	Actor	A party that manages a CSMS.
Electric Vehicle	Device	Electric Vehicle, distributed energy resource with a remote battery and socket.
Local Controller	Device	A logical entity between a CSMS and one or more Charging Stations that has the ability to control charging of a group of Charging Stations based on the input from the CSMS.
External Control System	Actor	An external system that may impose charging limits/constraints on the Charging Station or CSMS, for example a DSO or EMS.

2.5. References

2.5.1. Generic references

Table 7. References

Reference	Description
[DNP3]	Distributed Network Protocol. https://www.dnp.org/About/Overview-of-DNP3-Protocol
[eMI3-BO]	"eMI3 standard version V1.0" http://emi3group.com/documents-links/
[IEC60870-5-104]	Set of standards which define systems used for telecontrol (supervisory control and data acquisition) in electrical engineering and power system automation applications. https://webstore.iec.ch/publication/3755
[IEC61850-7-420]	Communications standard for distributed energy resources (DER). https://webstore.iec.ch/publication/6019
[IEC61851-1]	"IEC 61851-1 2017: EV conductive charging system - Part 1: General requirements" https://webstore.iec.ch/publication/33644
[IEC62196]	IEC 62196: Plugs, socket-outlets, vehicle couplers and vehicle inlets - Conductive charging of electric vehicles. https://webstore.iec.ch/publication/6582
[ISO15118-1]	ISO 15118-1 specifies terms and definitions, general requirements and use cases as the basis for the other parts of ISO 15118. It provides a general overview and a common understanding of aspects influencing the charge process, payment and load leveling. https://webstore.iec.ch/publication/9272
[ISO15118-2]	Road vehicles – Vehicle to grid communication interface – Part 2: Technical protocol description and Open Systems Interconnection (OSI) layer requirements, Document Identifier: 69/216/CDV. https://webstore.iec.ch/publication/9273
[ISO4217]	"ISO 4217: Currency codes" http://www.iso.org/iso/home/standards/currency_codes.htm
[OCPP2.0-PART4]	"OCPP 2.0.1: Part 4 - JSON over WebSockets implementation guide". http://www.openchargealliance.org/downloads/
[OpenADR]	"Open Automated Demand Response" http://www.openadr.org/
[RFC1321]	"The MD5 Message-Digest Algorithm" https://tools.ietf.org/html/rfc1321
[RFC2119]	"Key words for use in RFCs to Indicate Requirement Levels". S. Bradner. March 1997. http://www.ietf.org/rfc/rfc2119.txt
[RFC3339]	"Date and Time on the Internet: Timestamps" https://tools.ietf.org/html/rfc3339
[RFC3986]	"Uniform Resource Identifier (URI): Generic Syntax" https://tools.ietf.org/html/rfc3986
[RFC5646]	"Tags for Identifying Languages" https://tools.ietf.org/html/rfc5646

2.5.2. Security related references

Table 8. Security related references

Reference	Description
[1]	ENISA European Network and Information Security Agency, Algorithms, key size and parameters report 2014, 2014. (last accessed on 17 January 2016) https://www.enisa.europa.eu/publications/algorithms-key-size-and-parameters-report-2014
[2]	National Institute of Standards and Technology. FIPS PUB 140-2, Security Requirements for Cryptographic Modules, May 2001. http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.140-2.pdf
[3]	Cooper, D., et al., Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, Internet Engineering Task Force, Request for Comments 5280, May 2008, http://www.ietf.org/rfc/rfc5280.txt
[4]	Dierks, T. and Rescorla, E., The Transport Layer Security (TLS) Protocol Version 1.2, Internet Engineering Task Force, Request for Comments 5246, August 2008, http://www.ietf.org/rfc/rfc5246.txt
[5]	Eastlake, D., Transport Layer Security (TLS) Extensions: Extension Definitions, Internet Engineering Task Force, Request for Comments 6066, January 2011, http://www.ietf.org/rfc/rfc6066.txt
[6]	McGrew, D. and Bailey, D., AES-CCM Cipher Suites for Transport Layer Security (TLS), Internet Engineering Task Force, Request for Comments 6655, July 2012, http://www.ietf.org/rfc/rfc6655.txt
[7]	Rescorla E. et al., Transport Layer Security (TLS) Renegotiation Indication Extension, Internet Engineering Task Force, Request for Comments 5746, February 2010, http://www.ietf.org/rfc/rfc5746.txt

Reference	Description
[8]	"Russel Housley, Tim Polk, Warwick Ford, and David Solo. Internet Public Key Infrastructure: X.509 Certificate and Certificate Revocation List (CRL) Profile, RFC 3280, April 2002." https://www.ietf.org/rfc/rfc3280.txt
[9]	Pettersen. "The Transport Layer Security (TLS) Multiple Certificate Status Request Extension." RFC 6961, June 2013. https://tools.ietf.org/html/rfc6961 .
[10]	Hollenbeck, S., "Transport Layer Security Protocol Compression Methods", RFC 3749, May 2004. https://www.ietf.org/rfc/rfc3749.txt
[11]	National Institute of Standards and Technology. Annex C: Approved Random Number Generators for FIPS PUB 140-2 [25], February 2012. https://csrc.nist.gov/csrc/media/publications/fips/140/2/final/documents/fips1402annexc.pdf
[12]	Bundesamt für Sicherheit in der Informationstechnik: Anwendungshinweise und Interpretationen zum Schema, AIS 20, Funktionalitätsklassen und Evaluationsmethodologie für deterministische Zufallszahlengeneratoren, Version 3.0, Bonn, Germany, May 2013. (in German) https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Zertifizierung/Interpretationen/AIS_20_pdf.html
[13]	Bundesamt für Sicherheit in der Informationstechnik: Anwendungshinweise und Interpretationen zum Schema, AIS 31, Funktionalitätsklassen und Evaluationsmethodologie für physikalische Zufallszahlengeneratoren, Version 3.0, Bonn, Germany, May 2013. (in German) https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Zertifizierung/Interpretationen/AIS_31_pdf.html
[14]	"OWASP - Transport Layer Protection Cheat Sheet. https://www.owasp.org/index.php/Transport_Layer_Protection_Cheat_Sheet#Extended_Validation_Certificates "
[15]	P. Hoffman and W.C.A. Wijngaards, Elliptic Curve Digital Signature Algorithm (DSA) for DNNSEC, Internet Engineering Task Force (IETF) RFC 6605, April 2012. http://www.ietf.org/rfc/rfc6605.txt
[16]	Adams, C., Farrell, S., Kause, T., and T. Mononen, "Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP)", RFC 4210, September 2005. https://www.ietf.org/rfc/rfc4210.txt
[17]	National Institute of Standards and Technology. Special Publication 800-57 Part 1 Rev. 4, Recommendation for Key Management. January 2016. https://csrc.nist.gov/publications/detail/sp/800-57-part-1/rev-4/final
[18]	RFC 2617. HTTP Authentication: Basic and Digest Access Authentication. https://www.ietf.org/rfc/rfc2617.txt
[19]	RFC 5280. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. https://www.ietf.org/rfc/rfc5280.txt
[20]	OCPP 1.6. Interface description between Charging Station and CSMS. October 2015. http://www.openchargealliance.org/downloads/
[21]	Eekelen, M. van, Poll, E., Hubbers, E., Vieira, B., Broek, F. van den: An end-to-end security design for smart EV-charging for Enexis and ElaadNL by LaQuSo1. December 2, 2014. https://www.elaad.nl/smart-charging-end2end-security-design/
[22]	RFC 2986. PKCS #10: Certification Request Syntax Specification, Version 1.7. https://www.ietf.org/rfc/rfc2986.txt
[23]	RSA-PSS. https://tools.ietf.org/html/rfc8017
[24]	Santesson, et al. "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP" RFC 6960. June 2013. https://tools.ietf.org/html/rfc6960
[25]	RFC 2818. HTTP Over TLS. https://tools.ietf.org/html/rfc2818

2.6. Definition of Transaction

This section is informative.

To support as many business cases as possible, and to prevent too many messages being sent when not needed for certain business cases, OCPP 2.0.1 supports flexible configuration of the start and stop of a transaction. This makes it possible to define the start and stop of a transaction depending on market demands.

See: [Flexible transaction start/stop](#) for more information.

2.6.1. Transaction in relation to Energy Transfer Period

The [Energy Transfer Period](#) is a period of time during which energy is transferred between the EV and the EVSE. There MAY be multiple Energy Transfer Periods during a [Transaction](#).

Multiple Energy Transfer Periods can be separated by either:

- an EVSE-initiated suspense of transfer during which the EVSE does not offer energy transfer, or;
- an EV-initiated suspense of transfer during which the EV remains electrically connected to the EVSE, or;
- an EV-initiated suspense of transfer during which the EV is not electrically connected to the EVSE.

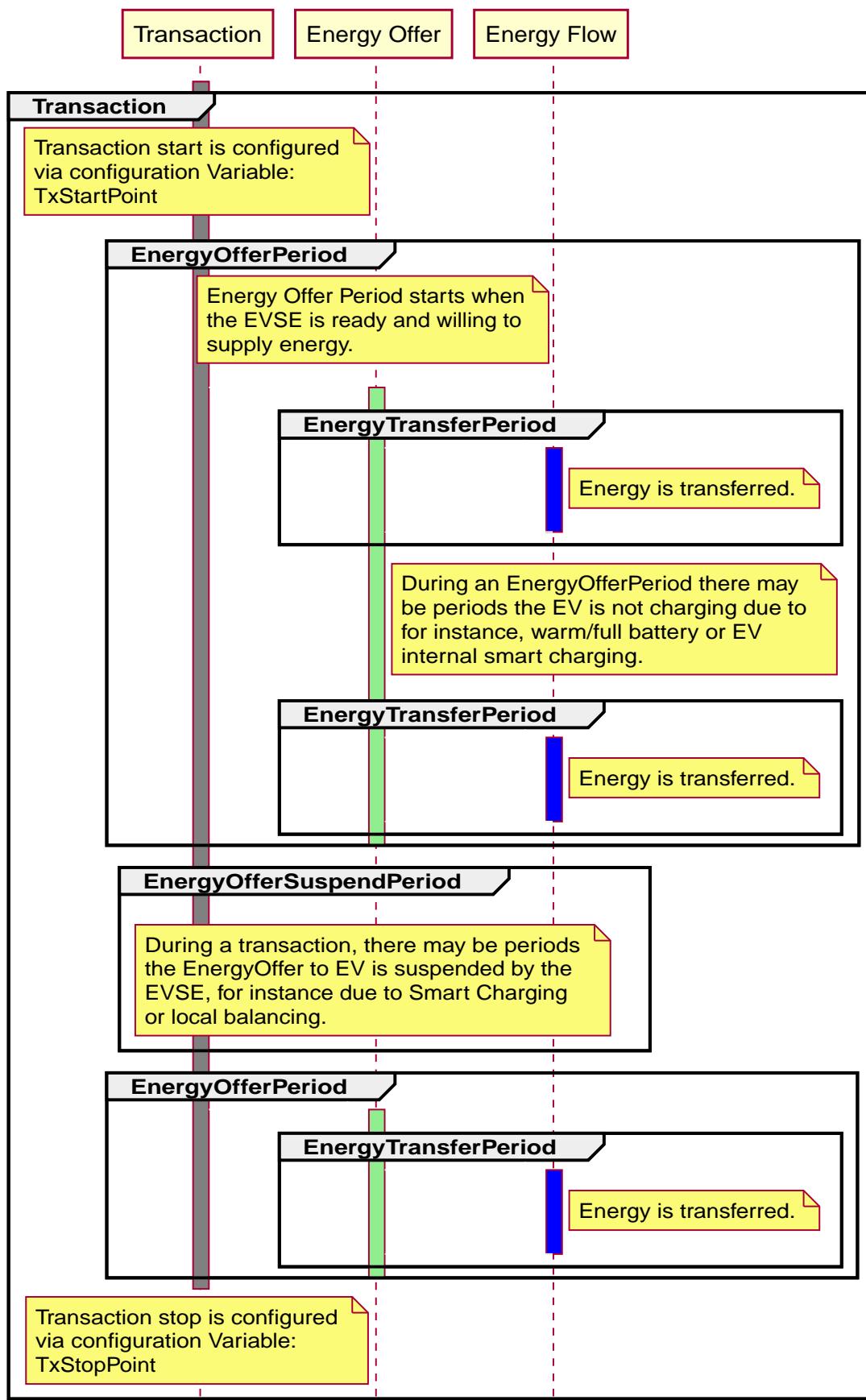


Figure 1. OCPP Charging Transaction definition

2.7. ISO 15118 support

This section is informative.

This version of OCPP supports ISO 15118 authorization (also called "Plug and Charge") and ISO 15118 based Smart Charging. (See [\[ISO15118-2\]](#)) Furthermore it describes how to install and update ISO 15118 certificates. These 3 functionalities are not included as one functional block, but are included in multiple chapters throughout the specification. ISO 15118 authorization is included in the functional block [Authorization](#) and the Smart Charging use cases for ISO 15118 are included in the chapter [Smart Charging](#). Certificate handling is described in a separate functional block.

Implementors of 15118 need to be aware of timeout constraints enforced by 15118, see [\[ISO15118-1\]](#) (Page: 127, Table: 109) For reference, the current timing constrains for 15118 edition 1 are:

Table 9. ISO 15118 Timing constrains

Timeout	Default
Sequence Timeouts	60 seconds
Sequence Performance Timeouts	40 seconds
PaymentDetailsReq/Res	5 seconds
CertificateUpdateReq/Res	5 seconds
CertificateInstallationReq/Res	5 seconds

3. Generic Requirements

This section is normative.

The generic requirements build the basis for defining the use case elements described in the Functional Blocks.

Table 10. Generic requirements

ID	Precondition	Requirement definition	Note
FR.01		The sender of a <message>Request SHALL wait for a <message>Response or a timeout, before sending another request message.	
FR.02	When the Charging Station receives a valid OCPP request message according to the JSON schemas / RPC Framework AND the other system is not causing a security violation	The Charging Station SHALL respond with a RPC Framework: CALLRESULT.	If the Charging Station/CSMS needs to provide additional information, this can be done in the <code>statusInfo</code> element of the response message.
FR.03	When the Charging Station/CSMS receives an invalid OCPP message according to the JSON schemas / RPC Framework OR the other system causes a security violation	The Charging Station/CSMS SHALL respond with a RPC Framework: CALLERROR.	
FR.04	When the CSMS did not accept the BootNotificationRequest from the Charging Station AND The Charging Station sends a message other than BootNotificationRequest	The CSMS SHALL respond with a RPC Framework: CALLERROR: SecurityError.	
FR.05	There are a few messages that do not provide their result in the response message, but send one or more messages that contain the result. When one of the following messages is received; GetReport, GetBaseReport, GetMonitoringReport, GetDisplayMessages, CustomerInformation, GetChargingProfiles, GetLog, UpdateFirmware, PublishFirmware, TriggerMessage(<message>)	The Charging Station SHALL acknowledge the requests in the list below with a response message first, before sending the follow-up message shown after the arrow (→): GetReport → NotifyReport GetBaseReport → NotifyReport GetMonitoringReport → NotifyMonitoringReport GetDisplayMessages → NotifyDisplayMessage CustomerInformation → NotifyCustomerInformation GetChargingProfiles → ReportChargingProfiles GetLog → LogStatusNotification UpdateFirmware → FirmwareStatusNotification PublishFirmware → PublishFirmwareStatusNotification TriggerMessage(<message>) → <requested message>	The CSMS needs to know that a request was accepted, so that it can expect result messages.

3.1. Time Format Requirements

This section is normative.

All time values exchanged between CSMS and Charging Station SHALL be formatted as defined in RFC-3339 [[RFC3339](#)]. Additionally fractional seconds have been given an extra limit. The number of decimal places SHALL NOT exceed the maximum of

3. However, it is RECOMMENDED to omit fractional seconds entirely, because it is of limited use and omitting it reduces data usages.

It is strongly RECOMMENDED to exchange all time values between CSMS and Charging Station as UTC, with the time zone designator 'Z', as specified by RFC-3339 [[RFC3339](#)]. This will improve interoperability between CSMS and Charging Station.

3.1.1. Displaying local time

When a Charging Station wants to give detailed control of configuring the internal clock to a CSO, it can implement one or more of the following Configuration Variables: [TimeSource](#), [TimeZone](#), [TimeOffset](#), [NtpSource](#), [NtpServerUri](#).

3.1.1.1. Daylight Saving Time

There are 2 ways a Charging Station can support punctual automated bi-annual changeover between "standard time" and "daylight saving time" periods.

- The transition dates and offsets are known in the Charging Station, based on the configured [TimeZone](#).
- The transition date and offset is manually configured for every transition via: [NextTimeOffsetTransitionDateTime](#) and [TimeOffsetNextTransition](#).

Daylight saving time is used for displaying the current time to the EV driver.

3.2. Message Timeouts

This section is normative.

OCPP does not specify timing requirements for messages. Timing of messages is greatly influenced by the underlying network used. A GPRS network has different timing characteristics compared to a land-line. As OCPP does not require a certain type of network, but leaves this open for the CSO to select, OCPP cannot require timing constraints.

If you are looking for some guidance, start with a 30 second timeout on message requests, and tune it for the network used.

The message timeout setting in a Charging Station can be configured in the `messageTimeout` field in the [NetworkConnectionProfile](#). The purpose of the message timeout is to be able to consider a request message as not sent and continue with other tasks when the message did not arrive due to communication errors or software failure. For transaction related events, use case [E13 - Transaction-related message not accepted by CSMS](#) describes the retry procedure when this happens. See also the section [Delivering transaction-related messages](#) in Functional Block E.

A charging station may discover that the connection to CSMS is not functioning correctly when it gets a timeout to a request or when the websocket ping is not answered. In such a situation it is advised that the charging station drops the connection and then reconnects to CSMS. This will create a fresh session and will possibly connect to a different endpoint of a multi-instance CSMS, which may resolve the error.

3.3. Language support

This section is informative.

A CSMS can provide the Charging Station with preferred languages for an EV Driver, enabling the Charging Station to communicate with the EV Driver in a language according to his/her preferences.

For any Charging Station that shows messages on a display it is RECOMMENDED to at least also implement these in "English". When the preferred languages for an EV-driver (provided by the CSMS) are not "English" and don't match any of the other languages implemented in the Charging Station, it is RECOMMENDED to use "English" as fall-back.

A. Security

1. OCPP Security

This Functional Block describes the security requirements for the OCPP protocol. The security part was developed to strengthen and mature the future development and standardization of OCPP. It is based amongst others on the end-to-end security design by LaQuSo [21]. Security requirements are included on security measures at Charging Station and CSMS, to support users of the OCPP.

1.1. Security Objectives

This section is informative.

OCPP security has been designed to meet the following security objectives:

1. To allow the creation of a secure communication channel between the CSMS and Charging Station. The integrity and confidentiality of messages on this channel should be protected with strong cryptographic measures.
2. To provide mutual authentication between the Charging Station and the CSMS. Both parties should be able to identify who they are communicating with.
3. To provide a secure firmware update process by allowing the Charging Station to check the source and the integrity of firmware images, and by allowing non-repudiation of these images.
4. To allow logging of security events to facilitate monitoring the security of the smart charging system. A list of security related events and their 'criticality' is provided in the appendices.

1.2. Design Considerations

This section is informative.

The security Functional Block was designed to fit into the approach taken in OCPP. Standard web technologies are used whenever possible to allow cost-effective implementations using available web libraries and software. No application layer security measures are included. Based on these considerations, OCPP security is based on TLS and public key cryptography using X.509 certificates. Because the CSMS usually acts as the server, different users or role-based access control on the Charging Station are not implemented in this standard. To mitigate this, it is recommended to implement access control on the CSMS. To make sure the mechanisms implemented there cannot be bypassed, OCPP should not be used by qualified personnel performing maintenance to Charging Stations locally at the Charging Station, as other protocols may be used for local maintenance purposes.

1.3. Security Profiles

This section defines the different OCPP security profiles and their requirement. OCPP 2.0.1 supports three security profiles: The table below shows which security measures are used by which profile.

Table 11. Overview of OCPP security profiles

Profile	Charging Station Authentication	CSMS Authentication	Communication Security
1. Unsecured Transport with Basic Authentication	HTTP Basic Authentication	-	-
2. TLS with Basic Authentication	HTTP Basic Authentication	TLS authentication using certificate	Transport Layer Security (TLS)
3. TLS with Client Side Certificates	TLS authentication using certificate	TLS authentication using certificate	Transport Layer Security (TLS)

- The [Unsecured Transport with Basic Authentication Profile](#) does not include authentication for the CSMS, or measures to set up a secure communication channel. Therefore, it should only be used in trusted networks, for instance in networks where there is a VPN between the CSMS and the Charging Station. For field operation it is highly recommended to use a security profile with TLS.
- In some cases (e.g. lab installations, test setups, etc.) one might prefer to use OCPP 2.0.1 without implementing security. While this is possible, it is NOT considered a valid OCPP 2.0.1 implementation.
- When the Charging Station does not have the correct date and time set, it cannot validate the server certificate. A solution for this might be to either use NTP, mobile network to set time automatically, or have an installer tool that sets the time before the first connection.

1.3.1. Generic Security Profile requirements

Table 12. Generic Security Profile requirements

ID	Precondition	Requirement definition
A00.FR.001		The Charging Station and CSMS SHALL only use one security profile at a time
A00.FR.002	If the Charging Station tries to connect with a different profile than the CSMS is using	The CSMS SHALL terminate the connection.
A00.FR.003	If the CSMS tries to connect with a different profile than the Charging Station is using	The Charging Station SHALL terminate the connection.
A00.FR.004		The security profile SHALL be configured before OCPP communication is possible.
A00.FR.005		Lowering the security profile that is used, to a less secure profile, is for security reasons, not part of the OCPP specification, and MUST be done through another method, not via OCPP. OCPP messages SHALL NOT be used for this (e.g. SetVariablesRequest or DataTransferRequest).
A00.FR.006	When a CSMS communicates with Charging Stations with different security profiles or different versions of OCPP.	The CSMS MAY operate the Charging Stations via different addresses or ports of the CSMS. For instance, the CSMS server may have one TCP port for TLS with Basic Authentication, and another port for TLS with Client Side Certificates. In this case there is only one security profile in use per port of the CSMS, which is allowed.

1.3.2. Unsecured Transport with Basic Authentication Profile - 1

Table 13. Security Profile 1 - Unsecured Transport with Basic Authentication

No.	Type	Description
1	Name	Unsecured Transport with Basic Authentication
2	Profile No.	1

No.	Type	Description
3	Description	The Unsecured Transport with Basic Authentication profile provides a low level of security. Charging Station authentication is done through a username and password. No measures are included to secure the communication channel.
4	Charging Station Authentication	For Charging Station authentication HTTP Basic authentication is used.
5	CSMS Authentication	In this profile, the CSMS does not authenticate itself to the Charging Station. The Charging Station has to trust that the server it connects to is indeed the CSMS.
6	Communication Security	No communication security measures are included in the profile.

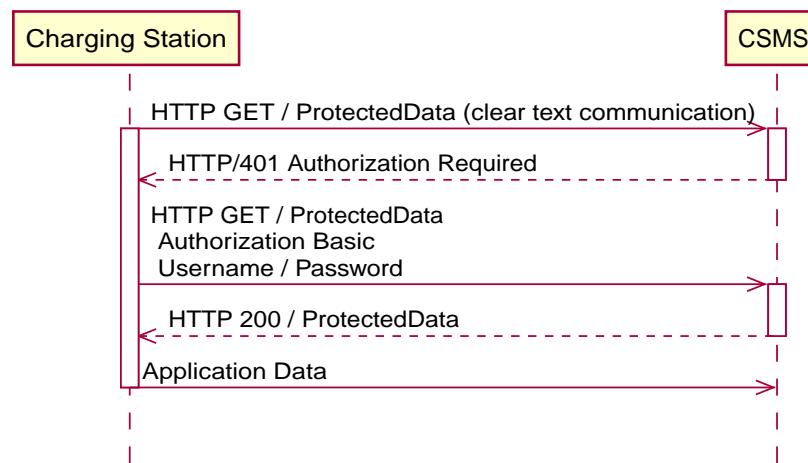


Figure 2. Sequence Diagram: HTTP Basic Authentication sequence diagram

7	Remark(s)	Please note, that the encoding of the basic authentication password in OCPP 2.0.1 (A00.FR.205) differs from how this was done in OCPP 1.6.
---	-----------	--

1.3.3. Unsecured Transport with Basic Authentication Profile - Requirements

Table 14. Security Profile 1 - Unsecured Transport with Basic Authentication - Requirements

ID	Precondition	Requirement definition
A00.FR.201		The Unsecured Transport with Basic Authentication Profile SHOULD only be used in trusted networks.
A00.FR.202		The Charging Station SHALL authenticate itself to the CSMS using HTTP Basic authentication [18]
A00.FR.203	A00.FR.202	The client, i.e. the Charging Station, SHALL provide a username and password with every connection request.
A00.FR.204	A00.FR.203	The username SHALL be equal to the Charging Station identity, which is the identifying string of the Charging Station as it uses it in the OCPP-J connection URL. When using Basic Authentication, the Charging Station identity may not contain the character ":". Otherwise the CSMS may be unable to separate the username from the password.
A00.FR.205		The password SHALL be stored in the BasicAuthPassword Configuration Variable. It SHALL be a randomly chosen passwordString with a sufficiently high entropy, consisting of minimum 16 and maximum 40 characters (alpha-numeric characters and the special characters allowed by passwordString). The password SHALL be sent as a UTF-8 encoded string (NOT encoded into octet string or base64).
A00.FR.206	A00.FR.203	With HTTP Basic, the username and password are transmitted in clear text, encoded in base64 only. Hence, it is RECOMMENDED that this mechanism will only be used over connections that are already secured with other means, such as VPNs.
A00.FR.207	A00.FR.202	The CSMS SHALL validate that Charging Station identity and the Basic Authentication password match with username and password in the authorization header of the connection request.

1.3.4. TLS with Basic Authentication Profile - 2

Table 15. Security Profile 2 - TLS with Basic Authentication

No.	Type	Description
1	Name	TLS with Basic Authentication
2	Profile No.	2
3	Description	In the TLS with Basic Authentication profile, the communication channel is secured using Transport Layer Security (TLS). The CSMS authenticates itself using a TLS server certificate. The Charging Stations authenticate themselves using HTTP Basic Authentication.
4	Charging Station Authentication	For Charging Station authentication HTTP Basic authentication is used. Because TLS is used in this profile, the password will be sent encrypted, reducing the risks of using this authentication method.
5	CSMS Authentication	The Charging Station authenticates the CSMS via the TLS server certificate.
6	Communication Security	The communication between Charging Station and CSMS is secured using TLS.

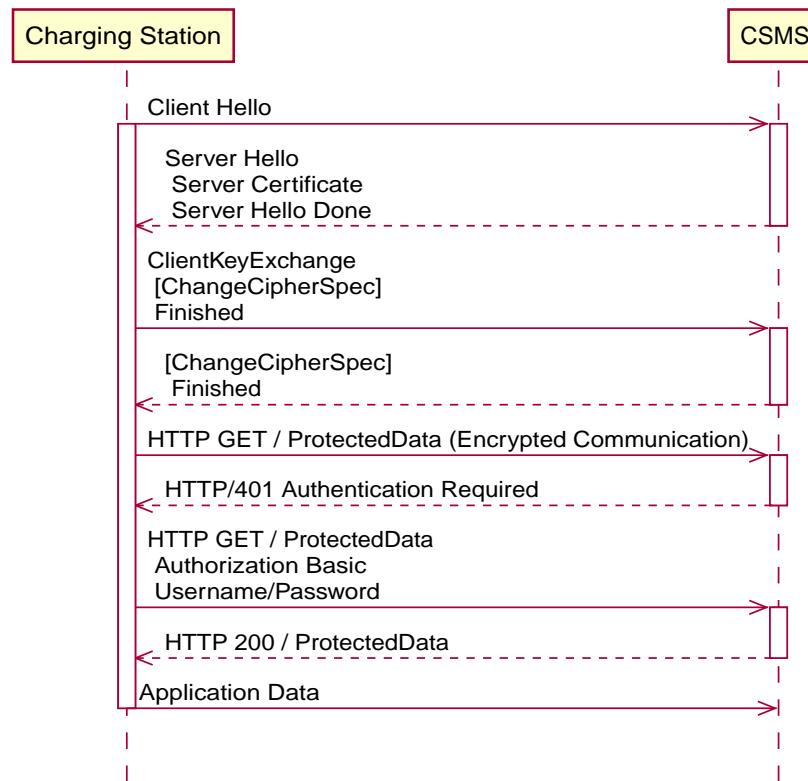


Figure 3. Sequence Diagram: TLS with Basic Authentication sequence diagram

7	Remark(s)	<p>TLS allows a number of configurations, not all of which provide sufficient security. The requirements below describe the configurations allowed for OCPP.</p> <p>The Charging Station should include the same header as used in Basic Auth RFC 2617, while requesting to upgrade the http connection to a websocket connection as described in RFC 6455. The server first needs to validate the Authorization header before upgrading the connection.</p> <p>Example:</p> <pre>GET /ws HTTP/1.1 Remote-Addr: 127.0.0.1 UPGRADE: websocket CONNECTION: Upgrade HOST: 127.0.0.1:9999 ORIGIN: http://127.0.0.1:9999 SEC-WEBSOCKET-KEY: Pb4obWo2214EfaPQuazMjA== SEC-WEBSOCKET-VERSION: 13 AUTHORIZATION: Basic <Base64 encoded(<ChargePointId>:<AuthorizationKey>)></pre> <p>Please note, that the encoding of the basic authentication password in OCPP 2.0.1 (A00.FR.304) differs from how this was done in OCPP 1.6.</p>
---	------------------	--

1.3.5. TLS with Basic Authentication Profile - Requirements

Table 16. Security Profile 2 - TLS with Basic Authentication - Requirements

ID	Precondition	Requirement definition
A00.FR.301		The Charging Station SHALL authenticate itself to the CSMS using HTTP Basic authentication [18]
A00.FR.302	A00.FR.301	The client, i.e. the Charging Station, SHALL provide a username and password with every connection request.
A00.FR.303	A00.FR.302	The username SHALL be equal to the Charging Station identity, which is the identifying string of the Charging Station as it uses it in the OCPP-J connection URL. When using Basic Authentication, the Charging Station identity may not contain the character ":". Otherwise the CSMS may be unable to separate the username from the password.
A00.FR.304	A00.FR.302	The password SHALL be stored in the <code>BasicAuthPassword</code> Configuration Variable. It SHALL be a randomly chosen passwordString with a sufficiently high entropy, consisting of minimum 16 and maximum 40 characters (alpha-numeric characters and the special characters allowed by passwordString). The password SHALL be sent as a UTF-8 encoded string (NOT encoded into octet string or base64).
A00.FR.306		The CSMS SHALL act as the TLS server.
A00.FR.307		The CSMS SHALL authenticate itself by using the CSMS certificate as server side certificate.
A00.FR.308		The Charging Station SHALL verify the certification path of the CSMS's certificate according to the path validation rules established in Section 6 of [3].
A00.FR.309		The Charging Station SHALL verify that the <code>commonName</code> includes the CSMS's FQDN.
A00.FR.310	If the CSMS does not own a valid certificate, or if the certification path is invalid	The Charging Station SHALL trigger an <code>InvalidCsmsCertificate</code> security event (See part 2 appendices for the full list of security events).
A00.FR.311	A00.FR.310	The Charging Station SHALL terminate the connection.
A00.FR.312		The communication channel SHALL be secured using Transport Layer Security (TLS) [4].
A00.FR.313		The Charging Station and CSMS SHALL only use TLS v1.2 or above.
A00.FR.314		Both of these endpoints SHALL check the version of TLS used.

ID	Precondition	Requirement definition
A00.FR.315	A00.FR.314 AND The CSMS detects that the Charging Station only allows connections using an older version of TLS, or only allows SSL	The CSMS SHALL terminate the connection.
A00.FR.316	A00.FR.314 AND The Charging Station detects that the CSMS only allows connections using an older version of TLS, or only allows SSL	The Charging Station SHALL trigger an InvalidTLSVersion security event AND terminate the connection (See part 2 appendices for the full list of security events).
A00.FR.317		TLS SHALL be implemented as in [4] or its successor standards without any modifications.
A00.FR.318		<p>The CSMS SHALL support at least the following four cipher suites:</p> <p>TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 TLS_RSA_WITH_AES_128_GCM_SHA256 TLS_RSA_WITH_AES_256_GCM_SHA384</p> <p>Note: The CSMS will have to provide 2 different certificates to support both cipher suites. Also when using security profile 3, the CSMS should be capable of generating client side certificates for both cipher suites.</p>
A00.FR.319		<p>The Charging Station SHALL support at least the cipher suites:</p> <p>(TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256) AND (TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384) OR (TLS_RSA_WITH_AES_128_GCM_SHA256) AND (TLS_RSA_WITH_AES_256_GCM_SHA384)</p> <p>Note 1: TLS_RSA does not support forward secrecy, therefore TLS_ECDHE is RECOMMENDED. Furthermore, if the Charging Station detects an algorithm used that is not secure, it SHOULD trigger an InvalidTLSCipherSuite security event (See part 2 appendices for the full list of security events).</p> <p>Note 2: Please note that ISO15118-2 prescribes to implement the following cipher suites for the communication between EV and Charging Station:</p> <p>TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256, TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256</p>
A00.FR.320		The Charging Station and CSMS SHALL NOT use cipher suites that use cryptographic primitives marked as unsuitable for legacy use in [1]. This will mean that when one (or more) of the cipher suites described in this specification becomes marked as unsuitable for legacy use, it SHALL NOT be used anymore.
A00.FR.321		The TLS Server and Client SHALL NOT use TLS compression methods to avoid compression side-channel attacks and to ensure interoperability as described in Section 6 of [10].
A00.FR.322	A00.FR.320 AND The CSMS detects that the Charging Station only allows connections using one of these suites	The CSMS SHALL terminate the connection.

ID	Precondition	Requirement definition
A00.FR.323	A00.FR.320 AND The Charging Station detects that the CSMS only allows connections using one of these suites	The Charging Station SHALL trigger an InvalidTLSCipherSuite security event AND terminate the connection (See part 2 appendices for the full list of security events).
A00.FR.324	A00.FR.302	The CSMS SHALL validate that Charging Station identity and the Basic Authentication password match with username and password in the authorization header of the connection request.

1.3.6. TLS with Client Side Certificates Profile - 3

Table 17. Security Profile 3 - TLS with Client Side Certificates

No.	Type	Description
1	Name	TLS with Client Side Certificates
2	Profile No.	3
3	Description	In the TLS with Client Side Certificates profile, the communication channel is secured using Transport Layer Security (TLS). Both the Charging Station and CSMS authenticate themselves using certificates.
4	Charging Station Authentication	The CSMS authenticates the Charging Station via the TLS client certificate.
5	CSMS Authentication	The Charging Station authenticates the CSMS via the TLS server certificate.
6	Communication Security	The communication between Charging Station and CSMS is secured using TLS.

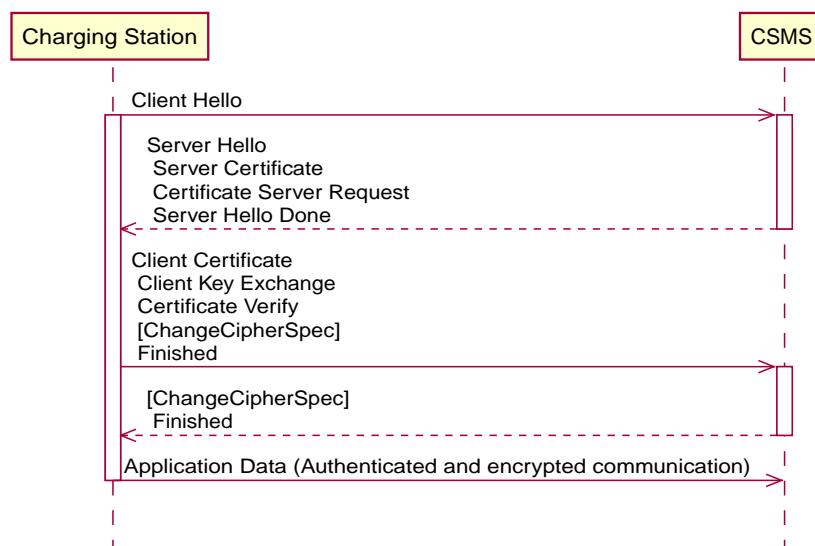


Figure 4. Sequence Diagram: TLS with Client Side Certificates

7	Remark(s)	N/a
---	-----------	-----

1.3.7. TLS with Client Side Certificates Profile - Requirements

Table 18. Security Profile 3 - TLS with Client Side Certificates - Requirements

ID	Precondition	Requirement definition
A00.FR.401		The Charging Station SHALL authenticate itself to the CSMS using the Charging Station certificate.
A00.FR.402		The Charging Station certificate SHALL be used as a TLS client side certificate
A00.FR.403		The CSMS SHALL verify the certification path of the Charging Station's certificate according to the path validation rules established in Section 6 of [3]

ID	Precondition	Requirement definition
A00.FR.404		The CSMS SHALL verify that the certificate is owned by the CSO (or an organization trusted by the CSO) by checking that the O (organizationName) RDN in the subject field of the certificate contains the CSO name.
A00.FR.405		The CSMS SHALL verify that the certificate belongs to this Charging Station by checking that the CN (commonName) RDN in the subject field of the certificate contains the unique serial number of the Charging Station (see Certificate Properties).
A00.FR.406	If the Charging Station certificate is not owned by the CSO, for instance immediately after installation	it is RECOMMENDED to update the certificate before continuing communication with the Charging Station (also see Installation)
A00.FR.407	NOT A00.FR.429 AND If the Charging Station does not own a valid certificate, or if the certification path is invalid	The CSMS SHALL terminate the connection.
A00.FR.408	A00.FR.407 OR A00.FR.429	It is RECOMMENDED to log a security event InvalidChargingStationCertificate in the CSMS.
A00.FR.409		The CSMS SHALL act as the TLS server.
A00.FR.410		The CSMS SHALL authenticate itself by using the CSMS certificate as server side certificate.
A00.FR.411		The Charging Station SHALL verify the certification path of the CSMS's certificate according to the path validation rules established in Section 6 of [3].
A00.FR.412		The Charging Station SHALL verify that the commonName matches the CSMS's FQDN.
A00.FR.413	If the CSMS does not own a valid certificate, or if the certification path is invalid	The Charging Station SHALL trigger an InvalidCsmsCertificate security event (See part 2 appendices for the full list of security events).
A00.FR.414	A00.FR.413	The Charging Station SHALL terminate the connection.
A00.FR.415		The communication channel SHALL be secured using Transport Layer Security (TLS) [4].
A00.FR.416		The Charging Station and CSMS SHALL only use TLS v1.2 or above.
A00.FR.417		Both of these endpoints SHALL check the version of TLS used.
A00.FR.418	A00.FR.417 AND The CSMS detects that the Charging Station only allows connections using an older version of TLS, or only allows SSL	The CSMS SHALL terminate the connection.
A00.FR.419	A00.FR.417 AND The Charging Station detects that the CSMS only allows connections using an older version of TLS, or only allows SSL	The Charging Station SHALL trigger an InvalidTLSVersion security event AND terminate the connection (See part 2 appendices for the full list of security events).
A00.FR.420		TLS SHALL be implemented as in [4] or its successor standards without any modifications.
A00.FR.421		The CSMS SHALL support at least the following four cipher suites: TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 TLS_RSA_WITH_AES_128_GCM_SHA256 TLS_RSA_WITH_AES_256_GCM_SHA384 Note: The CSMS will have to provide 2 different certificates to support both cipher suites. Also when using security profile 3, the CSMS should be capable of generating client side certificates for both cipher suites.

ID	Precondition	Requirement definition
A00.FR.422		<p>The Charging Station SHALL support at least the cipher suites: (TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256) AND TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384) OR (TLS_RSA_WITH_AES_128_GCM_SHA256) AND TLS_RSA_WITH_AES_256_GCM_SHA384)</p> <p>Note 1: TLS_RSA does not support forward secrecy, therefore TLS_ECDHE is RECOMMENDED. Furthermore, if the Charging Station detects an algorithm used that is not secure, it SHOULD trigger an InvalidTLSCipherSuite security event (See part 2 appendices for the full list of security events).</p> <p>Note 2: Please note that ISO15118-2 prescribes to implement the following cipher suites for the communication between EV and Charging Station:</p> <p>TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256, TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256</p>
A00.FR.423		The Charging Station and CSMS SHALL NOT use cipher suites that use cryptographic primitives marked as unsuitable for legacy use in [1] . This will mean that when one (or more) of the cipher suites described in this specification becomes marked as unsuitable for legacy use, it SHALL NOT be used anymore.
A00.FR.424		The TLS Server and Client SHALL NOT use TLS compression methods to avoid compression side-channel attacks and to ensure interoperability as described in Section 6 of [10] .
A00.FR.425	A00.FR.424 AND If the CSMS detects that the Charging Station only allows connections using one of these suites	The CSMS SHALL terminate the connection.
A00.FR.426	A00.FR.424 AND The Charging Station detects that the CSMS only allows connections using one of these suites	The Charging Station SHALL trigger an InvalidTLSCipherSuite security event AND terminate the connection (See part 2 appendices for the full list of security events).
A00.FR.427		A unique Charging Station certificate SHALL be used for each Charging Station.
A00.FR.428		The Charging Station Certificate MAY be the same certificate as the SECC Certificate in ISO15118-2 , used to set up a TLS connection between the Charging Station and an Electric Vehicle.
A00.FR.429	If Charging Station certificate has been expired AND CSMS has been explicitly configured to accept a connection by this specific Charging Station with an expired certificate.	CSMS MAY accept this Charging Station in a BootNotification - Pending state (use case B02) after which it SHALL immediately execute A02 - Update Charging Station Certificate by request of CSMS to renew the certificate.

1.4. Keys used in OCPP

This section is normative.

OCPP uses a number of public private key pairs for its security, see below Table. To manage the keys on the Charging Station, messages have been added to OCPP. Updating keys on the CSMS or at the manufacturer is out of scope for OCPP. If TLS with Client Side certificates is used, the Charging Station requires a "Charging Station certificate" for authentication against the CSMS.

Table 19. Certificates used in the OCPP security specification

Certificate	Private Key Stored At	Description
CSMS Certificate	CSMS	Key used to authenticate the CSMS.
Charging Station Certificate	Charging Station	Key used to authenticate the Charging Station.
Firmware Signing Certificate	Manufacturer	Key used to verify the firmware signature.
SECC Certificate	Charging Station	Certificate used by ISO15118-2 to set up a TLS connection between the Charging Station and an Electric Vehicle.

1.4.1. Certificate Properties

This section is normative.

Table 20. Certificate Properties requirements

ID	Precondition	Requirement definition
A00.FR.501		All certificates SHALL use a private key that provides security equivalent to a symmetric key of at least 112 bits according to Section 5.6.1 of [17] . This is the key size that NIST recommends for the period 2011-2030.
A00.FR.502	A00.FR.501 AND RSA or DSA	This translates into a key that SHALL be at least 2048 bits long.
A00.FR.503	A00.FR.501 AND elliptic curve cryptography	This translates into a key that SHALL be at least 224 bits long.
A00.FR.504		For all cryptographic operations, only the algorithms recommended by BSI in [12] , which are suitable for use in future systems, SHALL be used. This restriction includes the signing of certificates in the certificate hierarchy
A00.FR.505		For signing by the certificate authority RSA-PSS, or ECDSA SHOULD be used.
A00.FR.506		For computing hash values the SHA256 algorithm SHOULD be used.
A00.FR.507		The certificates SHALL be stored and transmitted in the X.509 format encoded in Privacy-Enhanced Mail (PEM) format.
A00.FR.508		All certificates SHALL include a serial number.
A00.FR.509		The subject field of the certificate SHALL contain the organization name of the certificate owner in the O (organizationName) RDN.
A00.FR.510		For the CSMS certificate, the subject field SHALL contain the FQDN of the endpoint of the server in the CN (commonName) RDN.
A00.FR.511		For the Charging Station certificate, the subject field SHALL contain a CN (commonName) RDN which consists of the unique serial number of the Charging Station. This serial number SHALL NOT be in the format of a URL or an IP address so that Charging Station certificates can be differentiated from CSMS certificates. Note: According to RFC 2818 , if a subjectAltName extension of type dnsName is present, that must be used as the identity. This would be incompliant with OCPP and ISO 15118 . Therefore it SHOULD NOT be used in Charging Station and CSMS certificates. It is allowed to use the subjectAltName extension of type dnsName for a CSMS, when the CSMS has multiple network paths to reach it (for example, via a private APN + VPN using its IP address in the VPN and via public Internet using a named URL).
A00.FR.512		For all certificates the X.509 Key Usage extension [19] SHOULD be used to restrict the usage of the certificate to the operations for which it will be used.
A00.FR.513		If the Charging Station Certificate is also used as SECC Certificate in the ISO 15118 protocol, the certificate SHOULD also meet the requirements in ISO15118-2 .

ID	Precondition	Requirement definition
A00.FR.514		For all certificates it is strongly RECOMMENDED NOT to use the X.509 Extended Key Usage extension, to be compatible with the ISO 15118 standard. There are alternative mechanisms available.

1.4.2. Certificate Hierarchy

This section is normative.

The OCPP protocol supports the use of two separate certificate hierarchies:

1. The Charging Station Operator hierarchy which contains the CSMS, and Charging Station certificates.
2. The Manufacturer hierarchy which contains the Firmware Signing certificate.

The CSMS can update the CSO root certificates stored on the Charging Station using the [InstallCertificateRequest](#) message.

Table 21. Certificate Hierarchy requirements

ID	Precondition	Requirement definition
A00.FR.601		The Charging Station Operator MAY act as a certificate authority for the Charging Station Operator hierarchy
A00.FR.602	A00.FR.601	The Charging Station Operator MAY for instance follow the certificate hierarchy described in Appendices E and F of ISO15118-2 and use the CSO Sub-CA 2 certificate to sign the CSMS and Charging Station certificates. This could give the advantage that the online verification of Charging Station client side certificates can be done within the Charging Station Operator's networks, simplifying the network architecture.
A00.FR.603		The private keys belonging to the CSO root certificates MUST be well protected.
A00.FR.604		As the Manufacturer is usually a separate organization from the Charging Station Operator, a trusted third party SHOULD be used as a certificate authority. This is essential to have non-repudiation of firmware images.

1.5. Certificate Revocation

This section is normative.

In some cases a certificate may become invalid prior to the expiration of the validity period. Such cases include changes of the organization name, or the compromise or suspected compromise of the certificate's private key. In such cases, the certificate needs to be revoked or indicate it is no longer valid. The revocation of the certificate does not mean that the connection needs to be closed as the the connection can stay open longer than 24 hours.

Different methods are recommended for certificate revocation, see below Table.

Table 22. Recommended revocation methods for the different certificates.

Certificate	Revocation
CSMS certificate	Fast expiration
Charging Station certificate	Online verification
Firmware Signing certificate	Online verification

Table 23. Certificate Revocation requirements

ID	Precondition	Requirement definition
A00.FR.701		Fast expiration SHOULD be used to revoke the CSMS certificate. (See Note 1)
A00.FR.702		The CSMS SHOULD use online certificate verification to verify the validity of the Charging Station certificates.
A00.FR.703		It is RECOMMENDED that a separate certificate authority server is used to manage the certificates.
A00.FR.704	A00.FR.703	This server SHOULD also keep track of which certificates have been revoked.

ID	Precondition	Requirement definition
A00.FR.705		The CSMS SHALL verify the validity of the certificate with the certificate authority server. (See Note 2)
A00.FR.707		Prior to providing the certificate for firmware validation to the Charging Station, the CSMS SHOULD validate both, the certificate and the signed firmware update.

Note 1: With fast expiration, the certificate is only valid for a short period, less than 24 hours. After that the server needs to request a new certificate from the Certificate Authority, which may be the CSO itself (see section [Certificate Hierarchy](#)). This prevents the Charging Stations from needing to implement revocation lists or online certificate verification. This simplifies the implementation of certificate management at the Charging Station and reduces communication costs at the Charging Station side. By requiring fast expiration, if the certificate is compromised, the impact is reduced to only a short period.

When the certificate chain becomes compromised, attackers could use forged certificates to trick a Charging Station to connect to a "fake" CSMS. By using fast expiration, the time a Charging Station is vulnerable is greatly reduced.

The Charging Station always communicates with the Certificate Authority through the CSMS, this way, if the Charging Station is compromised, the Charging Station cannot attack the CA directly.

Note 2: This allows for immediate revocation of Charging Station certificates. Revocation of Charging Station certificates will happen for instance when a Charging Station is removed. This is more common than revoking the CSMS certificate, which is normally only done when it is compromised.

1.6. Installation

This section is normative.

Unique credentials should be used to authenticate each Charging Station to the CSMS, whether they are the password used for HTTP Basic Authentication (see [Charging Station Authentication](#)) or the Charging Station certificate. These unique credentials have to be put on the Charging Station at some point during manufacturing or installation.

Table 24. Certificate Installation requirements

ID	Precondition	Requirement definition
A00.FR.801		It is RECOMMENDED that the manufacturer initializes the Charging Station with unique credentials during manufacturing.
A00.FR.802	A00.FR.801	The credentials SHOULD be generated using a cryptographic random number generator, and installed in a secure environment.
A00.FR.803	A00.FR.801	They SHOULD be sent to the CSO over a secure channel, so that the CSO can import them in the CSMS
A00.FR.804	If Charging Station certificates are used.	The manufacturer MAY sign these using their own certificate.
A00.FR.805	A00.FR.804	It is RECOMMENDED that the CSO immediately updates the credentials after installation using the methods described in Section A01 - Update Charging Station Password for HTTP Basic Authentication or A02 - Update Charging Station Certificate by request of CSMS .
A00.FR.806	Before the 'factory credentials' have been updated	The CSMS MAY restrict the functionality that the Charging Station can use. The CSMS can use the BootNotification state: Pending for this. During the Pending state, the CSMS can update the credentials.
A00.FR.807	A00.FR.804 AND Charging Station manufacturer certificate has expired	The CSMS MAY accept a connection by Charging Station in a Pending state after the BootNotification and immediately execute use case A02 - Update Charging Station Certificate by request of CSMS to install a new valid CSO certificate.

2. Use cases & Requirements

A01 - Update Charging Station Password for HTTP Basic Authentication

Table 25. A01 - Password Management

No.	Type	Description
1	Name	Update Charging Station Password for HTTP Basic Authentication
2	ID	A01
	Functional block	A. Security
3	Objective(s)	This use case defines how to use the BasicAuthPassword, the password used to authenticate Charging Stations in the Basic and TLS with Basic Authentication security profiles.
4	Description	To enable the CSMS to configure a new password for HTTP Basic Authentication, the CSMS can send a new value for the BasicAuthPassword Configuration Variable.
	Actors	Charging Station, CSMS
	Scenario description	<p>1. The CSMS sends a SetVariablesRequest(ComponentName=SecurityCtrlr, VariableName=BasicAuthPassword) to the Charging Station.</p> <p>2. The Charging Station responds with SetVariablesResponse and the status Accepted.</p> <p>3. The Charging Station disconnects its current connection. (Storing any queued messages)</p> <p>4. The Charging Station connects to the CSMS with the new password.</p>
5	Prerequisite(s)	Security Profile: Basic Security Profile or TLS with Basic Authentication in use.
6	Postcondition(s)	<p>Successful postcondition: The Charging Station has reconnected to the CSMS with the new password.</p> <p>Failure postcondition: If the Charging Station responds to the SetVariablesRequest with a SetVariablesResponse with a status other than Accepted, the Charging Station will keep using the old credentials. The CSMS might treat the Charging Station differently, e.g. by not accepting the Charging Station's boot notifications.</p>

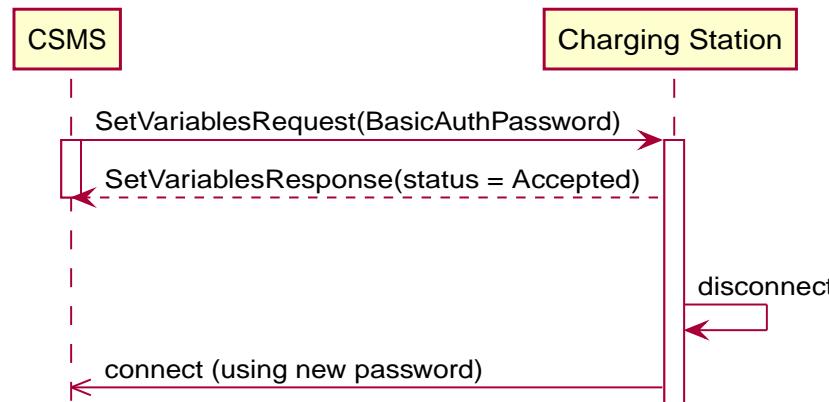


Figure 5. Update Charging Station Password for HTTP Basic Authentication (happy flow)

7	Error handling	n/a
8	Remark(s)	n/a

A01 - Update Charging Station Password for HTTP Basic Authentication - Requirements

Table 26. A01 - Update Charging Station Password for HTTP Basic Authentication - Requirements

ID	Precondition	Requirement definition
A01.FR.01		The password SHALL be stored in the configuration variable BasicAuthPassword .

ID	Precondition	Requirement definition
A01.FR.02		To set a Charging Station's basic authorization password via OCPP, the CSMS SHALL send the Charging Station a SetVariablesRequest message with the BasicAuthPassword Configuration Variable.
A01.FR.03	A01.FR.02 AND The Charging Station responds to this SetVariablesRequest with a SetVariablesResponse with status Accepted .	The CSMS SHALL assume that the authorization key change was successful, and no longer accept the credentials previously used by the Charging Station.
A01.FR.04	A01.FR.02 AND The Charging Station responds to this SetVariablesRequest with a SetVariablesResponse with status other than Accepted	The CSMS SHALL assume that the Charging Station has NOT changed the password. Therefore the CSMS SHALL keep accepting the old credentials.
A01.FR.05	A01.FR.04	While the CSMS SHALL still accept a connection from the Charging Station, it MAY restrict the functionality that the Charging Station can use. The CSMS can use the BootNotification state: Pending for this. During the Pending state, the CSMS can for example retry to update the credentials.
A01.FR.06		Different passwords SHOULD be used for different Charging Stations.
A01.FR.07		Passwords SHOULD be generated randomly to ensure that the passwords have sufficient entropy.
A01.FR.08		the CSMS SHOULD only store salted password hashes, not the passwords themselves.
A01.FR.09		the CSMS SHOULD NOT put the passwords in clear-text in log files or debug information. In this way, if the CSMS is compromised not all Charging Station password will be immediately compromised.
A01.FR.10		On the Charging Station the password needs to be stored in clear-text. Extra care SHOULD be taken into storing it securely. Definitions of mechanisms how to securely store the credentials are however not in scope of the OCPP Security Profiles.
A01.FR.11	A01.FR.02	The Charging Station SHALL log the change of an BasicAuthPassword in the Security log.
A01.FR.12	A01.FR.11	The Charging Station SHALL NOT disclose the content of the BasicAuthPassword in its logging. This is to prevent exposure of key material to persons that may have access to a diagnostics file.

A02 - Update Charging Station Certificate by request of CSMS

Table 27. A02 - Update Charging Station Certificate by request of CSMS

No.	Type	Description
1	Name	Update Charging Station Certificate by request of CSMS
2	ID	A02
	<i>Functional block</i>	A. Security
3	Objective(s)	To facilitate the management of the Charging Station client side certificate, a certificate update procedure is provided.
4	Description	The CSMS requests the Charging Station to update its key using TriggerMessageRequest with the requestedMessage field set to SignChargingStationCertificate (or SignV2GCertificate for separate 15118 certificate).
	Actors	Charging Station, CSMS, Certificate Authority Server

No.	Type	Description
	Scenario description	<p>1. The CSMS requests the Charging Station to update its certificate using the TriggerMessageRequest with the <code>requestedMessage</code> field set to SignChargingStationCertificate (or SignV2GCertificate for separate 15118 certificate).</p> <p>2. The Charging Station responds with TriggerMessageResponse</p> <p>3. The Charging Station generates a new public / private key pair.</p> <p>4. The Charging Station sends a SignCertificateRequest to the CSMS containing the applicable CertificateSigningUse.</p> <p>5. The CSMS responds with SignCertificateResponse, with status <code>Accepted</code>.</p> <p>6. The CSMS forwards the CSR to the Certificate Authority Server.</p> <p>7. Certificate Authority Server signs the certificate.</p> <p>8. The Certificate Authority Server returns the Signed Certificate to the CSMS.</p> <p>9. The CSMS sends CertificateSignedRequest to the Charging Station.</p> <p>10. The Charging Station verifies the Signed Certificate.</p> <p>11. The Charging Station responds with CertificateSignedResponse to the CSMS with the status <code>Accepted</code> or <code>Rejected</code>.</p>
5	Prerequisite(s)	The standard configuration variable <code>OrganizationName</code> MUST be set.
6	Postcondition(s)	<p>Successful postcondition: New Client Side certificate installed in the Charging Station.</p> <p>Failure postcondition: New Client Side certificate is rejected and discarded.</p>

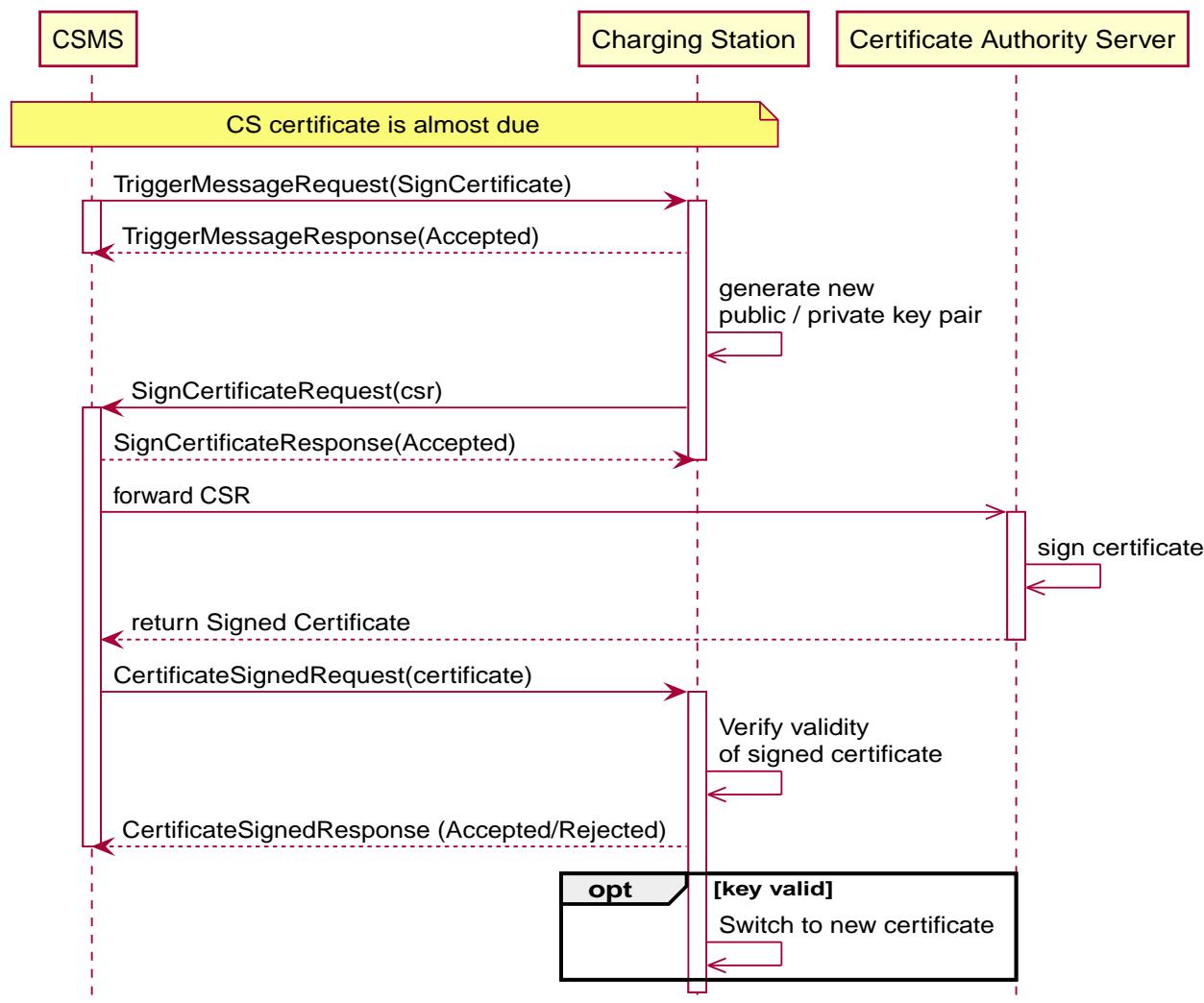


Figure 6. Update Charging Station Certificate

7	Error handling	The CSMS accepts the CSR request from the Charging Station, before forwarding it to the CA. But when the CA cannot be reached, or rejects the CSR, the Charging Station will never know. The CSMS may do some checks on the CSR, but cannot do all the checks that a CA does, and it does not prevent connection timeout to the CA. When something like this goes wrong, either the CA is offline or the CSR send by the Charging Station is not correct, according to the CA. In both cases this is something an operator at the CSO needs to be notified of. The operator then needs to investigate the issue. When resolved, the operator can re-run A02. It is NOT RECOMMENDED to let the Charging Station retry when the certificate is not send within X minutes or hours. When the CSR is incorrect, that will not be resolved automatically. It is possible that only a new firmware will fix this.
8	Remark(s)	<p>The Charging Station Operator may act as a certificate authority for the Charging Station Operator hierarchy.</p> <p>The applicable Certification Authority SHALL check the information in the CSR. If it is correct, the Certificate Authority SHALL sign the CSR, send it to the CSO, the CSO sends it back to the Charging Station in the CertificateSignedRequest message. The certificate authority SHOULD implement strong measures to keep the certificate signing private keys secure.</p> <p>Even though the messages CertificateSignedRequest (see use cases A02 and A03) and InstallCertificateRequest (use case M05 - Install CA Certificate in a Charging Station) are both used to send certificates, their purposes are different. CertificateSignedRequest is used to return the the Charging Stations own public certificate and V2G certificate(s) signed by a Certificate Authority. InstallCertificateRequest is used to install Root certificates.</p> <p>For V2G certificate handling see use cases M03 - Retrieve list of available certificates from a Charging Station, M04 - Delete a specific certificate from a Charging Station and M06 - Get Charging Station Certificate status.</p>

A02 - Update Charging Station Certificate by request of CSMS - Requirements

Table 28. A02 - Requirements

ID	Precondition	Requirement definition
A02.FR.01		A key update SHOULD be performed after installation of the Charging Station, to change the key from the one initially provisioned by the manufacturer (possibly a default key).
A02.FR.02	After sending a TriggerMessageResponse .	The Charging Station SHALL generate a new public / private key pair using one of the key generation functions described in Section 4.2.1.3 of [16].
A02.FR.03	A02.FR.02	The Charging Station SHALL send the public key in form of a Certificate Signing Request (CSR) as described in RFC 2986 [22] and then PEM encoded, using the SignCertificateRequest message.
A02.FR.04		The CSMS SHOULD NOT sign the certificate itself, but instead forwards the CSR to a dedicated certificate authority server managing the certificates for the Charging Station infrastructure. The dedicated authority server MAY be operated by the CSO.
A02.FR.05		The private key generated by the Charging Station during the key update process SHALL NOT leave the Charging Station at any time, and SHALL NOT be readable via OCPP or any other (remote) communication connection.
A02.FR.06		The Charging Station SHALL verify the validity of the signed certificate in the CertificateSignedRequest message, checking at least the period when the certificate is valid, the properties in Certificate Properties , and that it is part of the Charging Station Operator certificate hierarchy as described in Certificate Hierarchy .
A02.FR.07	If the certificate is not valid.	The Charging Station SHALL respond to the CertificateSignedRequest with status <i>Rejected</i> AND discard the certificate AND trigger an <i>InvalidChargingStationCertificate</i> security event (See part 2 appendices for the full list of security events).

ID	Precondition	Requirement definition
A02.FR.08		The Charging Station SHALL switch to the new certificate as soon as the current date and time is after the 'Not valid before' field in the certificate (e.g. by closing the websocket and TLS connection and reconnecting with the new certificate).
A02.FR.09	If the Charging Station contains more than one valid certificate of the <i>ChargingStationCertificate</i> type.	The Charging Station SHALL use the newest certificate, as measured by the start of the validity period.
A02.FR.10	A02.FR.09 AND When the Charging Station has validated that the new certificate works	The Charging Station MAY discard the old certificate. It is RECOMMENDED to store old certificates for one month, as fallback.
A02.FR.11	Upon receipt of a <i>SignCertificateRequest</i> AND It is able to process the request	The CSMS SHALL set status to Accepted in the <i>SignCertificateResponse</i> .
A02.FR.12	Upon receipt of a <i>SignCertificateRequest</i> AND It is NOT able to process the request	The CSMS SHALL set status to Rejected in the <i>SignCertificateResponse</i> .
A02.FR.13	When using different certificates for 15118 connections and the Charging Station to CSMS connection	The Charging Station SHALL set the <i>certificateType</i> field in the <i>SignCertificateRequest</i> to the certificate for which the update was triggered.
A02.FR.14	When receiving a <i>SignCertificateRequest</i> with <i>certificateType</i> included	It is RECOMMENDED for the CSMS to set the <i>certificateType</i> field in the <i>CertificateSignedRequest</i> to the type of certificate in the <i>SignCertificateRequest</i> .
A02.FR.15	If the Charging Station contains more than one valid V2G certificate, derived from the same root certificate.	The Charging Station SHALL use the newest certificate, as measured by the start of the validity period.
A02.FR.16	If the configuration variable <i>MaxCertificateChainSize</i> is implemented AND The Charging Station receives a <i>CertificateSignedRequest</i> message with a certificate (chain) with a size that exceeds the set value configured at <i>MaxCertificateChainSize</i>	The Charging Station SHALL respond with a <i>CertificateSignedResponse</i> message with status Rejected .
A02.FR.17	When the CSMS accepted the <i>SignCertificateRequest</i> for a CSR AND the Charging Station did not yet receive a <i>CertificateSignedRequest</i> for this CSR AND the number of seconds configured at <i>CertSigningWaitMinimum</i> has expired	The Charging Station SHALL send a new <i>SignCertificateRequest</i> for the CSR. Optionally, this CSR MAY be for a newly generated key pair.
A02.FR.18	A02.FR.17	The Charging Station SHALL double the previous back-off time, starting with the number of seconds configured at <i>CertSigningWaitMinimum</i> , every time the back-off time expires without having received the <i>CertificateSignedRequest</i> for this CSR.
A02.FR.19	A02.FR.18 AND The maximum number of increments is reached	The Charging Station SHALL stop resending the <i>SignCertificateRequest</i> , until it is requested by the CSMS via a <i>TriggerMessageRequest</i> for <i>SignChargingStationCertificate</i> , <i>SignV2GCertificate</i> or <i>SignCombinedCertificate</i> .
A02.FR.20	A02.FR.07	The Charging Station SHALL NOT initiate the back-off mechanism and resend the <i>SignCertificateRequest</i> , until this is requested by the CSMS via a <i>TriggerMessageRequest</i> for <i>SignChargingStationCertificate</i> , <i>SignV2GCertificate</i> or <i>SignCombinedCertificate</i> .

ID	Precondition	Requirement definition
A02.FR.21	When the Charging Station receives a SignCertificateResponse with status <i>Rejected</i> , in response to a SignCertificateRequest with certificateType <i>V2GCertificate</i>	It is RECOMMENDED to turn off ISO15118PnCEnabled until the Charging Station has been rebooted.

A03 - Update Charging Station Certificate initiated by the Charging Station

Table 29. A03 - Update Charging Station Certificate initiated by the Charging Station

No.	Type	Description
1	Name	Update Charging Station Certificate initiated by the Charging Station
2	ID	A03
	<i>Functional block</i>	A. Security
3	Objective(s)	To facilitate the management of the Charging Station client side certificate, a certificate update procedure is provided.
4	Description	The Charging Station detects that the certificate (ChargingStationCertificate or V2GCertificate for 15118) it is using will expire in one month. The Charging Station initiates the process to update its key using SignCertificateRequest , indicating the requested certificate in the CertificateSigningUse field.
	Actors	Charging Station, CSMS, Certificate Authority Server
	<i>Scenario description</i>	<ol style="list-style-type: none"> 1. The Charging Station detects that the Charging Station certificate is due to expire. 2. The Charging Station generates a new public / private key pair. 3. The Charging Station sends a SignCertificateRequest to the CSMS containing the applicable CertificateSigningUse. 4. The CSMS responds with a SignCertificateResponse, with status <i>Accepted</i>. 5. The CSMS forwards the CSR to the Certificate Authority Server. 6. Certificate Authority Server signs the certificate. 7. The Certificate Authority Server returns the Signed Certificate to the CSMS. 8. The CSMS sends a CertificateSignedRequest to the Charging Station. 9. The Charging Station verifies the Signed Certificate. 10. The Charging Station responds with a CertificateSignedResponse to the CSMS with the status <i>Accepted</i> or <i>Rejected</i>.
5	Prerequisite(s)	The standard configuration variable <code>OrganizationName</code> MUST be set.
6	Postcondition(s)	Successful postcondition: New Client Side certificate installed in the Charging Station. Failure postcondition: New Client Side certificate is rejected and discarded.

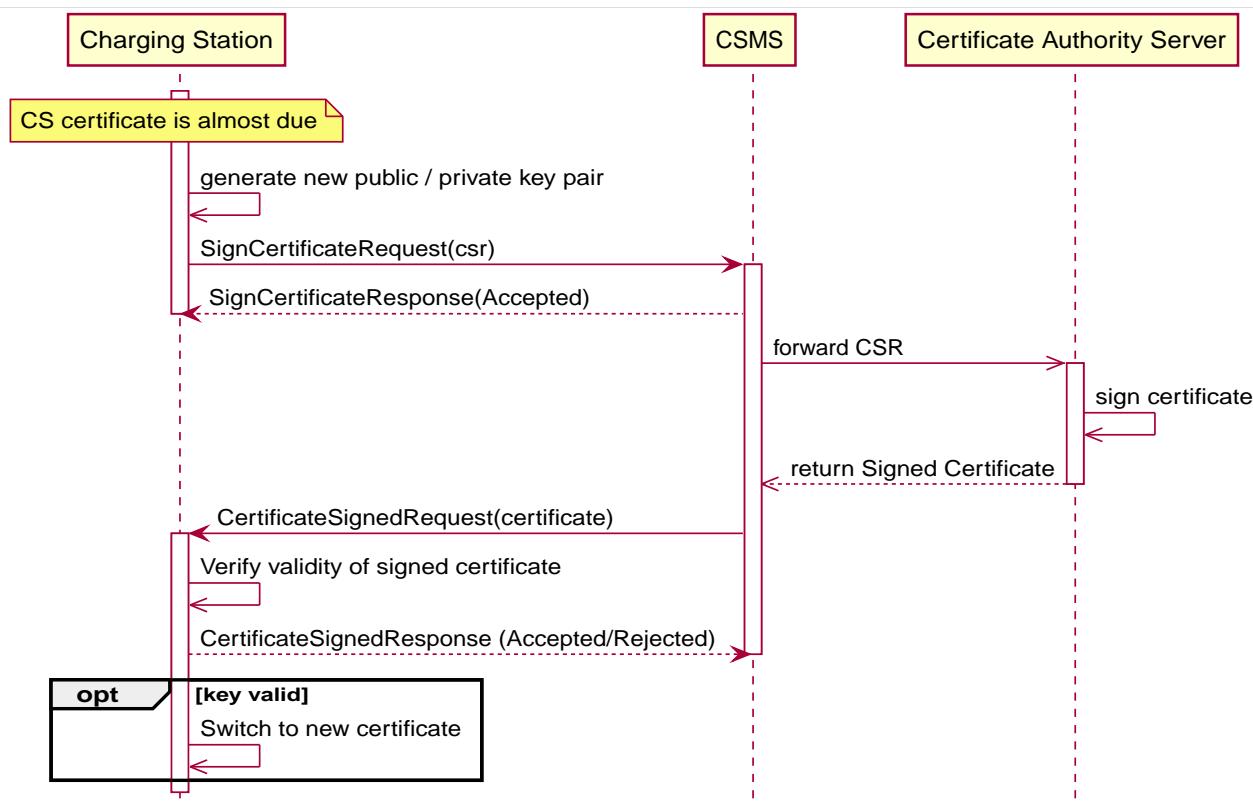


Figure 7. Update Charging Station Certificate initiated by Charging Station

7	Error handling	The CSMS accepts the CSR request from the Charging Station, before forwarding it to the CA. But when the CA cannot be reached, or rejects the CSR, the Charging Station will never know. The CSMS may do some checks on the CSR, but cannot do all the checks that a CA does, and it does not prevent connection timeout to the CA. When something like this goes wrong, either the CA is offline or the CSR send by the Charging Station is not correct, according to the CA. In both cases this is something an operator at the CSO needs to be notified of. The operator then needs to investigate the issue. When resolved, the operator can re-run A02. It is NOT RECOMMENDED to let the Charging Station retry when the certificate is not send within X minutes or hours. When the CSR is incorrect, that will not be resolved automatically. It is possible that only a new firmware will fix this.
8	Remark(s)	Same remarks as in A02 - Update Charging Station Certificate by request of CSMS apply.

A03 - Update Charging Station Certificate initiated by the Charging Station - Requirements

Table 30. A03 - Requirements

ID	Precondition	Requirement definition
A03.FR.01		A key update MAY be performed after installation of the Charging Station, to change the key from the one initially provisioned by the manufacturer (possibly a default key).
A03.FR.02	When the Charging Station detects that the current Charging Station certificate will expire in one month.	The Charging Station SHALL generate a new public / private key pair using one of the key generation functions described in Section 4.2.1.3 of [16].
A03.FR.03	A03.FR.02	The Charging Station SHALL send the public key in form of a Certificate Signing Request (CSR) as described in RFC 2986 [22] and then PEM encoded, using the SignCertificateRequest message.
A03.FR.04		The CSMS SHOULD NOT sign the certificate itself, but instead forwards the CSR to a dedicated certificate authority server managing the certificates for the Charging Station infrastructure. The dedicated authority server MAY be operated by the CSO.
A03.FR.05		The private key generated by the Charging Station during the key update process SHALL NOT leave the Charging Station at any time, and SHALL NOT be readable via OCPP or any other (remote) communication connection.

ID	Precondition	Requirement definition
A03.FR.06		The Charging Station SHALL verify the validity of the signed certificate in the CertificateSignedRequest message, checking at least the period when the certificate is valid, the properties in Certificate Properties , and that it is part of the Charging Station Operator certificate hierarchy as described in Certificate Hierarchy .
A03.FR.07	If the certificate is not valid.	The Charging Station SHALL respond to the CertificateSignedRequest with status Rejected AND discard the certificate AND trigger an InvalidChargingStationCertificate security event (See part 2 appendices for the full list of security events).
A03.FR.08		The Charging Station SHALL switch to the new certificate as soon as the current date and time is after the 'Not valid before' field in the certificate (e.g. by closing the websocket and TLS connection and reconnecting with the new certificate).
A03.FR.09	If the Charging Station contains more than one valid certificate of the ChargingStationCertificate type.	The Charging Station SHALL use the newest certificate, as measured by the start of the validity period.
A03.FR.10	A03.FR09 AND When the Charging Station has validated that the new certificate works	The Charging Station MAY discard the old certificate. It is RECOMMENDED to store old certificates for one month, as fallback.
A03.FR.11	Upon receipt of a SignCertificateRequest AND It is able to process the request	The CSMS SHALL set status to Accepted in the SignCertificateResponse .
A03.FR.12	Upon receipt of a SignCertificateRequest AND It is NOT able to process the request	The CSMS SHALL set status to Rejected in the SignCertificateResponse .
A03.FR.13	When using different certificates for 15118 connections and the Charging Station to CSMS connection	The Charging Station SHALL include the certificateType field in the SignCertificateRequest to specify which certificate it wants to update.
A03.FR.14	When receiving a SignCertificateRequest with certificateType included	It is RECOMMENDED for the CSMS to set the certificateType field in the CertificateSignedRequest to the type of certificate in the SignCertificateRequest .
A03.FR.15	If the Charging Station contains more than one valid V2G certificate, derived from the same root certificate.	The Charging Station SHALL use the newest certificate, as measured by the start of the validity period.
A03.FR.16	If the configuration variable MaxCertificateChainSize is implemented AND The Charging Station receives a CertificateSignedRequest message with a certificate (chain) with a size that exceeds the set value configured at MaxCertificateChainSize	The Charging Station SHALL respond with a CertificateSignedResponse message with status Rejected .
A03.FR.17	When the CSMS accepted the SignCertificateRequest for a CSR AND the Charging Station did not yet receive a CertificateSignedRequest for this CSR AND the number of seconds configured at CertSigningWaitMinimum has expired	The Charging Station SHALL send a new SignCertificateRequest for the CSR. Optionally, this CSR MAY be for a newly generated key pair.
A03.FR.18	A03.FR.17	The Charging Station SHALL double the previous back-off time, starting with the number of seconds configured at CertSigningWaitMinimum , every time the back-off time expires without having received the CertificateSignedRequest for this CSR.

ID	Precondition	Requirement definition
A03.FR.19	A03.FR.18 AND The maximum number of increments is reached	The Charging Station SHALL stop resending the SignCertificateRequest, until it is requested by the CSMS via a TriggerMessageRequest for SignChargingStationCertificate, SignV2GCertificate or SignCombinedCertificate.

A04 - Security Event Notification

Table 31. A04 - Security Event Notification

No.	Type	Description
1	Name	Security Event Notification
2	ID	A04
	Functional block	A. Security
3	Objective(s)	To inform the CSMS of critical security events.
4	Description	This use case allows the Charging Station to immediately inform the CSMS of changes in the system security.
	Actors	CSMS, Charging Station
	Scenario description	<ol style="list-style-type: none"> 1. A critical security event happens. 2. The Charging Station sends a SecurityEventNotificationRequest to the CSMS. 3. The CSMS responds with SecurityEventNotificationResponse to the Charging Station.
5	Prerequisite(s)	n/a
6	Postcondition(s)	The Charging Station successfully informs the CSMS of critical security events by sending a SecurityEventNotificationRequest to the CSMS.

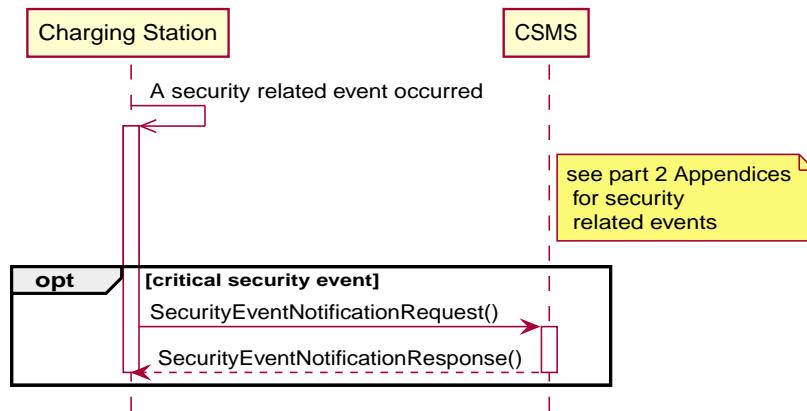


Figure 8. Security Event Notification

7	Error handling	n/a
8	Remark(s)	A list of security related events and their 'criticality' is provided in the Appendices (Appendix 1. Security Events)

A04 - Security Event Notification - Requirements

Table 32. A04 - Security Event Notification - Requirements

ID	Precondition	Requirement definition	Note
A04.FR.01	When a <i>critical</i> security event happens	The Charging Station SHALL inform the CSMS of the security events by sending a SecurityEventNotificationRequest to the CSMS.	
A04.FR.02	A04.FR.01 AND the Charging Station is disconnected.	Security event notifications MUST be queued with a guaranteed delivery at the CSMS.	
A04.FR.03	A04.FR.01	The CSMS SHALL confirm the receipt of the notification using the SecurityEventNotificationResponse message.	
A04.FR.04	When a security event happens (also non-critical)	The Charging Station SHALL store the security event in a security log.	It is recommended to implement this log in a rolling format.

A05 - Upgrade Charging Station Security Profile

Table 33. A05 - Upgrade Charging Station Security Profile

No.	Type	Description
1	Name	Upgrade Charging Station Security Profile
2	ID	A05
	Functional block	A. Security
3	Objective(s)	The CSO wants to increase the security of the OCPP connection between CSMS and a Charging Station.
4	Description	Use case when migrating from OCPP 1.6 without security profiles to OCPP 1.6 with security profiles or OCPP 2.0.1 Before migrating to a security profile the prerequisites, like installed certificates or password need to be configured.
	Actors	CSMS, Charging Station
	Scenario description	<p>1. The CSMS sets a new value for the <code>NetworkConfigurationPriority</code> Configuration Variable via <code>SetVariablesRequest</code>, such that the <code>NetworkConnectionProfile</code> for the new (higher) security profile becomes first in the list and the existing connection profile becomes second in the list.</p> <p>2. The Charging Station responds with a <code>SetVariablesResponse</code> with status Accepted</p> <p>3. The CSMS sends a <code>ResetRequest(OnIdle)</code></p> <p>4. The Charging Station reboots and connects via the new primary <code>NetworkConnectionProfile</code></p>
5	Prerequisite(s)	The CSO ensures that a <code>NetworkConnectionProfile</code> has been set using (higher) security profile AND that the prerequisite(s) for going to a higher security profile are met before sending the command to change to a higher security profile.
6	Postcondition(s)	The Charging Station was successfully upgraded to a higher security profile.

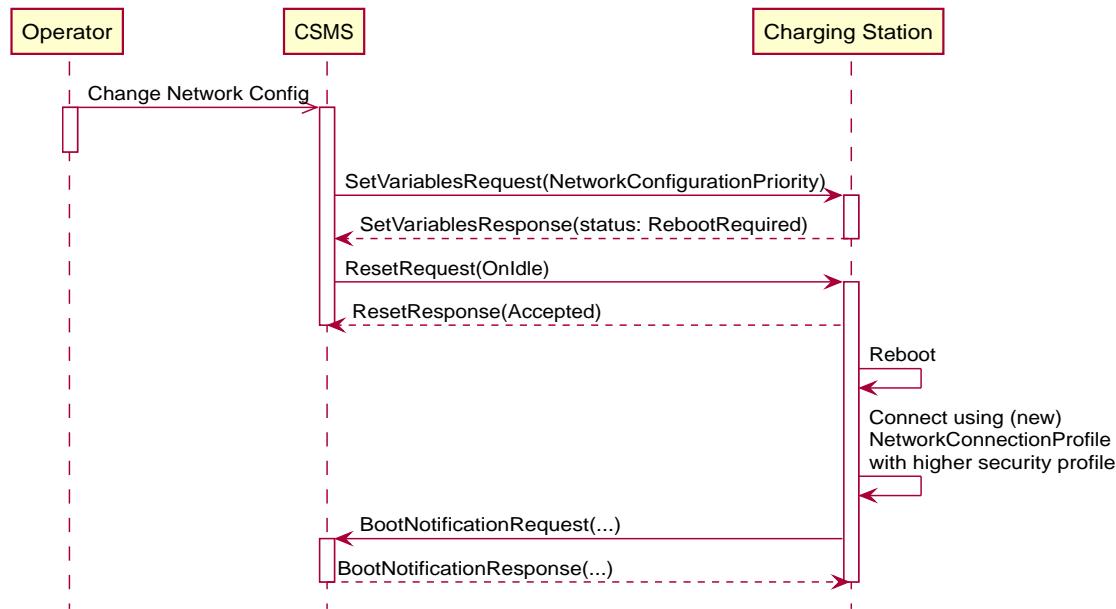


Figure 9. Upgrade Charging Station Security Profile

7	Error handling	n/a
8	Remark(s)	For security reasons it is not allowed to revert to a lower Security Profile using OCPP.

A05 - Upgrade Charging Station Security Profile - Requirements

Table 34. A05 - Upgrade Charging Station Security Profile

ID	Precondition	Requirement definition
A05.FR.02	<p>The Charging Station receives SetVariablesRequest for NetworkConfigurationPriority containing a profile slot for a NetworkConnectionProfile with a 'securityProfile' value higher than the current value AND new value is 2 or 3 AND No valid CSMSRootCertificate installed</p>	<p>The Charging Station SHALL respond with SetVariablesResponse(Rejected), and not update the value for SecurityProfile and/or reconnect to the CSMS.</p>
A05.FR.03	<p>The Charging Station receives SetVariablesRequest for NetworkConfigurationPriority containing a profile slot for a NetworkConnectionProfile with a 'securityProfile' value higher than the current value AND new value is 3 AND No valid ChargingStationCertificate installed</p>	<p>The Charging Station SHALL respond with SetVariablesResponse(Rejected), and not update the value for SecurityProfile and/or reconnect to the CSMS.</p>
A05.FR.04	<p>The Charging Station receives SetVariablesRequest for NetworkConfigurationPriority containing profile slots for NetworkConnectionProfiles with a 'securityProfile' value equal to or higher than the current value AND all prerequisites are met</p>	<p>The Charging Station SHALL respond with SetVariablesResponse(Accepted)</p>
A05.FR.05	A05.FR.04 AND After a reboot	The Charging Station SHALL begin connecting to the first entry of NetworkConfigurationPriority
A05.FR.06	A05.FR.05 AND The Charging Station successfully connected to the CSMS using the (new) NetworkConnectionProfile	<p>The Charging Station SHALL update the value of the configuration variable SecurityProfile AND it SHALL remove all NetworkConnectionProfiles with a lower securityProfile than stored at SecurityProfile AND update NetworkConfigurationPriority accordingly.</p>
A05.FR.07	A05.FR.06	The CSMS SHALL NOT allow the Charging Station to connect with a lower security profile anymore.

B. Provisioning

1. Introduction

This Functional Block describes all the functionalities that help a CSO provision their Charging Stations, allowing them on their network and retrieving configuration information from these Charging Stations. Additionally, it consists of the ability to retrieve information about the configuration of Charging Stations, make changes to the configuration etc. This chapter also covers resetting a Charging Station and migrating to a new NetworkConnectionProfile.

1.1. Transactions before being accepted by a CSMS

A Charging Station Operator MAY choose to configure a Charging Station to accept transactions before the Charging Station is accepted by a CSMS. Parties who want to implement this such behavior should realize that it is uncertain if those transactions can ever be delivered to the CSMS.

After a restart (for instance due to a remote reset command, power outage, firmware update, software error etc.) the Charging Station MUST again contact the CSMS and SHALL send a BootNotification request. If the Charging Station fails to receive a [BootNotificationResponse](#) from the CSMS, and has no in-built non-volatile real-time clock hardware that has been correctly preset, the Charging Station may not have a valid date and time setting, making it difficult or even impossible to later determine the date and time of transactions.

It might also be the case (e.g. due to configuration error) that the CSMS indicates a status other than Accepted for an extended period of time, or indefinitely.

It is usually advisable to deny all charging services at a Charging Station if the Charging Station has never before been Accepted by the CSMS (using the current connection settings, URL, etc.) since users cannot be authenticated and running transactions could conflict with provisioning processes.

If this is supported, this behaviour can be configured via the Configuration Variable: [TxBeforeAcceptedEnabled](#).

2. Use cases & Requirements

2.1. Booting a Charging Station

B01 - Cold Boot Charging Station

Table 35. B01 - Cold Boot Charging Station

No.	Type	Description
1	Name	Cold Boot Charging Station
2	ID	B01
	Functional block	B. Provisioning
3	Objective(s)	The objective of this use case is to enable a Charging Station that is powering up to register itself at a CSMS and provide the right state information.
4	Description	This use case describes how the CSMS can control which Charging Stations access it. To be able to control Charging Stations connecting to a CSMS, Charging Stations are required to send BootNotificationRequest . This request contains some general information about the Charging Station.
	Actors	Charging Station, CSMS
	Scenario description	<ol style="list-style-type: none"> 1. The Charging Station is powered up. 2. The Charging Station sends BootNotificationRequest to the CSMS. 3. The CSMS returns with BootNotificationResponse with the status Accepted. 4. <i>Optional:</i> The Charging Station sends StatusNotificationRequest with status <i>Unavailable</i> to the CSMS for each Connector. 5. The Charging Station sends StatusNotificationRequest to the CSMS for each Connector. If the status was set to <i>Unavailable</i> or <i>Reserved</i> from the CSMS prior to the (re)boot, the Connector should return to this status, otherwise the status should be <i>Available</i> or, when it resumes a transaction that was ongoing, the status should be <i>Occupied</i>. 6. Normal operational is resumed. 7. The Charging Station sends HeartbeatRequest to the CSMS.
	Alternative scenario(s)	B02 - Cold Boot Charging Station - Pending B03 - Cold Boot Charging Station - Rejected
5	Prerequisite(s)	The Charging Station is powered down.
6	Postcondition(s)	<p>Successful postcondition: The Charging Station is in <i>Idle</i> status, and Accepted.</p> <p>Failure postcondition: The Charging Station received the status <i>Rejected</i>, B03 - Cold Boot Charging Station -Rejected applies.</p> <p>The Charging Station received the status <i>Pending</i>, B02 - Cold Boot Charging Station - Pending applies.</p>

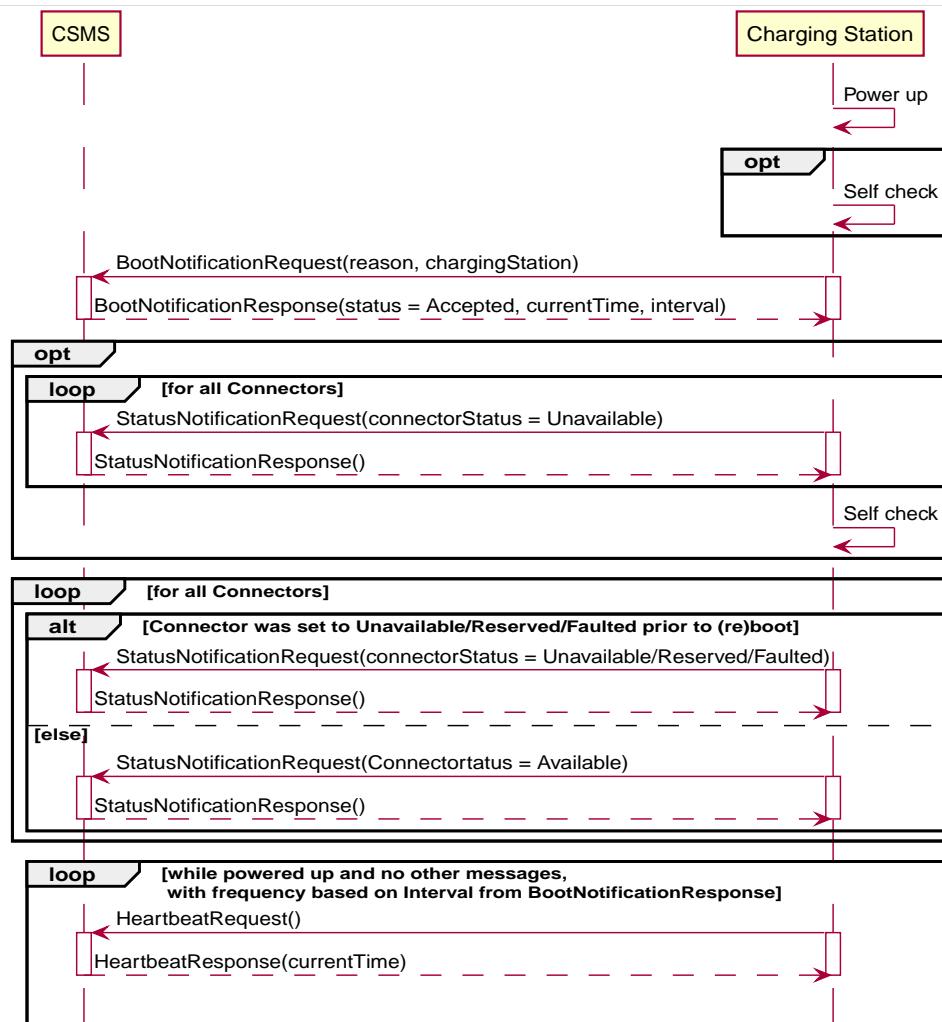


Figure 10. Sequence Diagram: Cold Boot Charging Station

7	Error handling	<p>1. No initial establishment of connection of communication between the CSMS and Charging Station: Retry Connection with the CSMS.</p> <p>2. No response / time-out from the CSMS: The Charging Station resends BootNotificationRequest after a waiting interval. The Charging Station chooses this interval on its own (since it did not get a BootNotificationResponse containing this interval), in a way that avoids flooding the CSMS with requests.</p>
8	Remark(s)	<p>Multiple options for a self check are possible: some Charging Stations boot and send status notifications with <i>Unavailable</i>, then perform a check of all the hardware and send new StatusNotifications with status <i>Available</i> when the Charging Station is up and running. However, there is no required order for a self check and sending a BootNotificationRequest. A Charging Stations can also do the self check <i>before</i> sending a BootNotificationRequest and determine the status before a (mobile) network connection is established and a BootNotificationRequest is sent.</p> <p>When something is wrong with the Charging Station or EVSE, the status SHALL be set to <i>Faulted</i>. <i>Reserved</i> and <i>Unavailable</i> states persist after a reboot.</p>

B01 - Cold Boot Charging Station - Requirements

Table 36. B01 - Requirements

ID	Precondition	Requirement definition	Note
B01.FR.01	After start-up.	The Charging Station SHALL send BootNotificationRequest to the CSMS with information about its configuration.	Information: e.g. version, vendor, etc.
B01.FR.02	B01.FR.01 The CSMS has received BootNotificationRequest from the Charging Station.	The CSMS SHALL respond to indicate whether it will accept the Charging Station.	

ID	Precondition	Requirement definition	Note
B01.FR.03	After a reboot (for instance due to a remote reset command, power outage, firmware update, software error etc.)	The Charging Station SHALL again connect to the CSMS and SHALL send a BootNotificationRequest each time it boots or reboots.	
B01.FR.04	When the CSMS responds with BootNotificationResponse with the status Accepted AND $interval > 0$	The Charging Station SHALL adjust the heartbeat interval in accordance with the interval from the response message.	
B01.FR.05	When the CSMS responds with BootNotificationResponse with the status Accepted.	The Charging Station SHALL send a StatusNotificationRequest for each Connector with its current state.	
B01.FR.06	The Charging Station has received BootNotificationResponse . AND Charging Station is configured to use Heartbeats for time synchronization TimeSource	The Charging Station SHALL synchronize the Charging Station's internal clock with the supplied CSMS's current time.	
B01.FR.07	When a Charging Station or an EVSE is set to status <i>Unavailable</i> by a Change Availability command.	The <i>Unavailable</i> status MUST be persistent across reboots.	
B01.FR.08	Between the physical power-on/reboot and the successful completion of a BootNotification, where the CSMS returns Accepted or Pending.	The Charging Station SHALL NOT send any other OCPP requests to the CSMS (Except BootNotificationRequest). This includes cached OCPP messages that are still present in the Charging Station from prior sessions.	Refer to B02 - Cold Boot Charging Station - Pending (for example B02.FR.02) for more details on sending messages on the <i>Pending</i> status.
B01.FR.09	B01.FR.01	The Charging Station SHALL indicate the reason for sending the BootNotificationRequest message in the <i>reason</i> field.	For which reason to use, see BootReasonEnumType .
B01.FR.10	The Charging Station has received a BootNotificationResponse in which status is not Accepted AND the Charging Station sends a RPC Framework: CALL message that is NOT a BootNotificationRequest or a message triggered by one of the following messages: TriggerMessageRequest , GetBaseReportRequest , GetReportRequest .	The CSMS SHALL respond with RPC Framework: CALLERROR: SecurityError.	The Charging Station is not allowed to initiate sending other messages before being accepted.
B01.FR.11	B01.FR.01 AND Security profile 3 is used	The CSMS SHALL check the SerialNumber in the BootNotificationRequest against the Serial Number in the Certificate Common Name.	
B01.FR.12	B01.FR.11 AND the SerialNumber in the BootNotificationRequest does NOT equal the Serial Number in the Certificate Common Name	The CSMS SHALL close WebSocket connection.	
B01.FR.13	When an EVSE has been reserved	The <i>Reserved</i> state MUST be persistent across reboots.	

B02 - Cold Boot Charging Station - Pending

Table 37. B02 - Cold Boot Charging Station - Pending

No.	Type	Description
1	Name	Cold Boot Charging Station - Pending
2	ID	B02
	Functional block	B. Provisioning
	Parent use case	B01 - Cold Boot Charging Station
3	Objective(s)	<ol style="list-style-type: none"> 1. To inform the Charging Station that it is not yet accepted by the CSMS: <i>Pending</i> status. 2. To give the CSMS a way to retrieve or set certain configuration information. 3. To give the CSMS a way of limiting the load on the CSMS after e.g. a reboot of the CSMS.
4	Description	This use case describes the behavior of the CSMS and a Charging Station when the Charging Station is informed by the CSMS that it is not yet accepted using the <i>Pending</i> status.
	Actors	Charging Station, CSMS
	Scenario description	<ol style="list-style-type: none"> 1. The Charging Station is powered up. 2. The Charging Station sends BootNotificationRequest to the CSMS. 3. The CSMS responds with BootNotificationResponse with the status <i>Pending</i>. 4. The CSMS then, is able to send messages to the Charging Station in order to change the configuration of the Charging Station. 5. The Charging Station resends BootNotificationRequest after the number of seconds indicated by the interval field. (Interval from BootNotificationResponse)
5	Prerequisite(s)	<ol style="list-style-type: none"> 1. The CSMS requires to set the Charging Station in <i>Pending</i> status. 2. The Charging Station is starting up (i.e. powering up after being powered down).
6	Postcondition(s)	<p>Successful postcondition: The Charging Station is in <i>Pending</i> status.</p> <p>Failure postcondition: The Charging Station received the status <i>Rejected</i>, B03 - Cold Boot Charging Station -Rejected applies.</p>

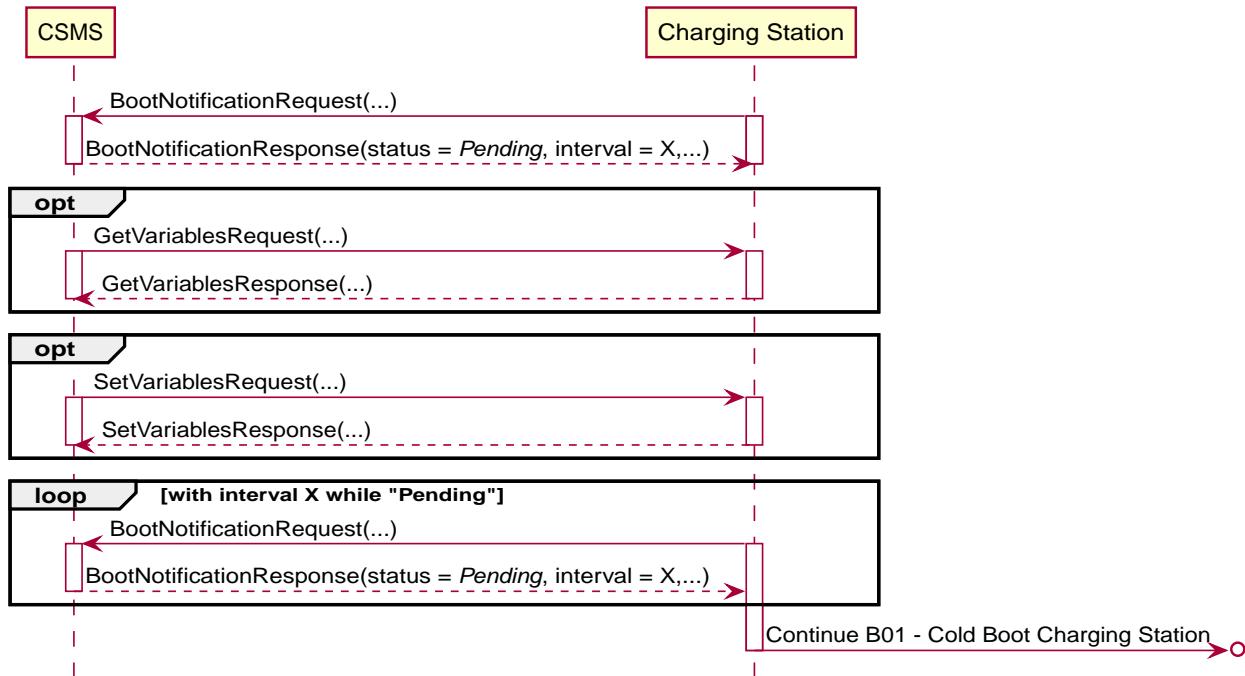


Figure 11. Sequence Diagram: Cold Boot Charging Station - Pending

7	Error handling	1. When no initial connection established between CSMS and Charging Station: Retry Connection to the CSMS and resend BootNotificationRequest . 2. No response / time-out from the CSMS: The Charging Station resends BootNotificationRequest after a waiting interval. This waiting interval can be based on the interval from a previous BootNotificationResponse or chosen by the Charging Station itself. In the latter case, the Charging Station chooses this interval in a way that avoids flooding the CSMS with requests.
8	Remark(s)	When the CSMS returns with BootNotificationResponse with the status Accepted, B01 - Cold Boot Charging Station applies.

B02 - Cold Boot Charging Station - Pending - Requirements

Table 38. B02 - Requirements

ID	Precondition	Requirement definition	Note
B02.FR.01	After the Charging Station received the <i>Pending</i> status.	The CSMS MAY send messages to retrieve information from the Charging Station (as described in use cases B06, B07, B08) or change its configuration by SetVariablesRequest (as described in use case B05). The Charging Station SHALL respond to these messages.	The Pending status can thus indicate that the CSMS wants to retrieve or set certain information on the Charging Station before it will accept the Charging Station.
B02.FR.02	While the CSMS has not yet responded to a BootNotificationRequest with an Accepted status in the BootNotificationResponse .	The Charging Station SHALL NOT send RPC Framework: CALL messages (Except BootNotificationRequest) to the CSMS, unless it has been instructed by the CSMS to do so, using one of the following messages: TriggerMessageRequest , GetBaseReportRequest , GetReportRequest .	
B02.FR.03	While the CSMS has not yet responded to a BootNotificationRequest with an Accepted status in the BootNotificationResponse .	A Charging Station Operator MAY choose to configure a Charging Station to accept transactions and queue TransactionEventRequest messages to be sent to the CSMS	Parties who want to implement this behavior must realize that it is uncertain if those transactions can ever be delivered to the CSMS.
B02.FR.04	While the CSMS has not yet responded to a BootNotificationRequest with an Accepted status in the BootNotificationResponse .	A Charging Station SHALL NOT send BootNotificationRequest earlier than the value of the Interval field in BootNotificationResponse , unless requested to do so with TriggerMessageRequest .	
B02.FR.05	While in <i>Pending</i> status AND receiving a RequestStartTransactionRequest or RequestStopTransactionRequest	The Charging Station SHALL respond with a RequestStartTransactionResponse or RequestStopTransactionResponse with status <i>Rejected</i> . (Even if the Charging Station is allowed to start transaction, see B02.FR.03. If the CSMS wants to use RequestStartTransaction etc. it SHALL first accept the Charging Station)	
B02.FR.06	When the CSMS returns the Pending status	The communication channel SHALL NOT be closed by either the Charging Station or the CSMS.	
B02.FR.07	If the interval in the BootNotificationResponse equals 0, and the status is other than Accepted,	The Charging Station SHALL choose a waiting interval on its own, in a way that avoids flooding the CSMS with requests.	
B02.FR.08	If the interval in the BootNotificationResponse > 0, and the status is other than Accepted,	The Charging Station SHALL send a BootNotificationRequest after the set interval has past.	

ID	Precondition	Requirement definition	Note
B02.FR.09	<p>The Charging Station has received a BootNotificationResponse with status <i>Pending</i></p> <p>AND</p> <p>the Charging Station sends a RPC Framework: CALL message that is NOT a BootNotificationRequest or a message triggered by one of the following messages:</p> <ul style="list-style-type: none"> TriggerMessageRequest, GetBaseReportRequest, GetReportRequest. 	<p>The CSMS SHALL respond with RPC Framework: CALLERROR: SecurityError.</p>	<p>The Charging Station is not allowed to initiate sending other messages before being accepted.</p>

B03 - Cold Boot Charging Station - Rejected

Table 39. B03 - Cold Boot Charging Station - Rejected

No.	Type	Description
1	Name	Cold Boot Charging Station - Rejected
2	ID	B03
	Functional block	B. Provisioning
	Parent use case	B01 - Cold Boot Charging Station
3	Objective(s)	To inform the Charging Station that its <i>not</i> (yet) accepted by the CSMS: Rejected status.
4	Description	This use case describes the behavior of the CSMS and a Charging Station, when the Charging Station is informed by the CSMS that it is not (yet) accepted using the <i>Rejected</i> status.
	Actors	Charging Station, CSMS
	Scenario description	<ol style="list-style-type: none"> 1. The Charging Station is powered up. 2. The Charging Station sends BootNotificationRequest to the CSMS. 3. The CSMS responds with BootNotificationResponse with the status <i>Rejected</i> to the Charging Station. 4. The Charging Station will resend BootNotificationRequest after the number of seconds indicated by the interval field. (Interval from BootNotificationResponse).
5	Prerequisite(s)	<ol style="list-style-type: none"> 1. The CSMS requires to set the Charging Station in the <i>Rejected</i> status. 2. The Charging Station is powered down.
6	Postcondition(s)	The Charging Station remains in the <i>Rejected</i> status.

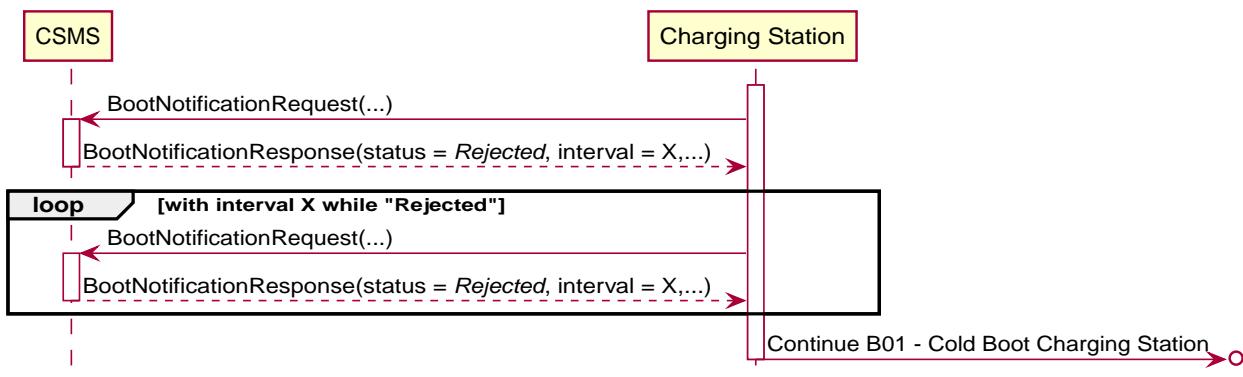


Figure 12. Sequence Diagram: Cold Boot Charging Station - Rejected

7	Error handling	<p>When there is no response or a time-out from the CSMS: The Charging Station resends BootNotificationRequest after a waiting interval. This waiting interval can be based on the interval from a previous BootNotificationResponse or chosen by the Charging Station itself. In the latter case, the Charging Station chooses this interval in a way that avoids flooding the CSMS with requests.</p>
8	Remark(s)	<p>During the status <i>Rejected</i>, the Charging Station may no longer be reachable from the CSMS. The Charging Station MAY e.g. close its communication channel or shut down its communication hardware.</p> <p>Additionally, the CSMS MAY close the communication channel, for instance to free up system resources.</p> <p>It is advised <i>not</i> to accept any transactions until the BootNotification of the Charging Station has been accepted by the CSMS. See: Transactions before being accepted by a CSMS</p> <p>When the CSMS returns with BootNotificationResponse with the status Accepted, B01 - Cold Boot Charging Station applies.</p>

B03 - Cold Boot Charging Station - Rejected - Requirements

Table 40. B03 - Requirements

ID	Precondition	Requirement definition
B03.FR.01	If the Charging Station is configured to accept Transactions before being accepted by a CSMS	The Charging Station MAY allow locally authorized transactions.
B03.FR.02	If the CSMS returns the status <i>Rejected</i> . For example when a Charging Station is blacklisted.	The Charging Station SHALL NOT send any OCPP message to the CSMS until the retry interval has expired.
B03.FR.03	While in the status <i>Rejected</i> .	The CSMS SHALL NOT initiate any messages.
B03.FR.04	B03.FR.03	The Charging Station MAY close the connection until it needs to send the next BootNotificationRequest .
B03.FR.05	If the interval in the BootNotificationResponse equals 0, and the status is other than Accepted	The Charging Station SHALL choose a waiting interval on its own, in a way that avoids flooding the CSMS with requests.
B03.FR.06	If the interval in the BootNotificationResponse is greater than 0, and the status is other than Accepted	The Charging Station SHALL send a BootNotificationRequest after the set interval has past.
B03.FR.07	B03.FR.03 AND Charging Station sends a message that is not a BootNotificationRequest	CSMS SHALL respond with RPC Framework: CALLERROR: SecurityError.
B03.FR.08	B03.FR.03 AND CSMS sends a message that is not a TriggerMessageRequest (requestedMessage = <i>BootNotification</i>)	Charging Station SHALL respond with RPC Framework: CALLERROR: SecurityError.

B04 - Offline Behavior Idle Charging Station

Table 41. B04 - Offline Behavior Idle Charging Station

No.	Type	Description
1	Name	Offline Behavior Idle Charging Station
2	ID	B04
	Functional block	B. Provisioning
3	Objective(s)	To attain stand-alone operation of the Charging Station.
4	Description	This use case describes that, in the event of unavailability of the communication, the Charging Station is designed to operate stand-alone. In that situation, the Charging Station is said to be <i>Offline</i> .
	Actors	Charging Station, CSMS
	Scenario description	<ol style="list-style-type: none"> 1. The CSMS or communication is unavailable. 2. The Charging Station operates stand-alone. 3. The connection is restored. 4. If the <i>Offline</i> period exceeds the value of the OfflineThreshold Configuration Variable: the Charging Station sends a StatusNotificationRequest to the CSMS for each connector. Otherwise it only sends a StatusNotificationRequest for Connectors with a status change during the offline period. 5. The Charging Station sends HeartbeatRequest to the CSMS. 6. The CSMS responds with HeartbeatResponse.
5	Prerequisite(s)	The BootNotification was previously accepted and the Charging Station is able to operate stand-alone.
6	Postcondition(s)	When connection is restored after a period of <i>Offline</i> behavior, the CSMS knows the Charging Stations' and EVSEs' state.

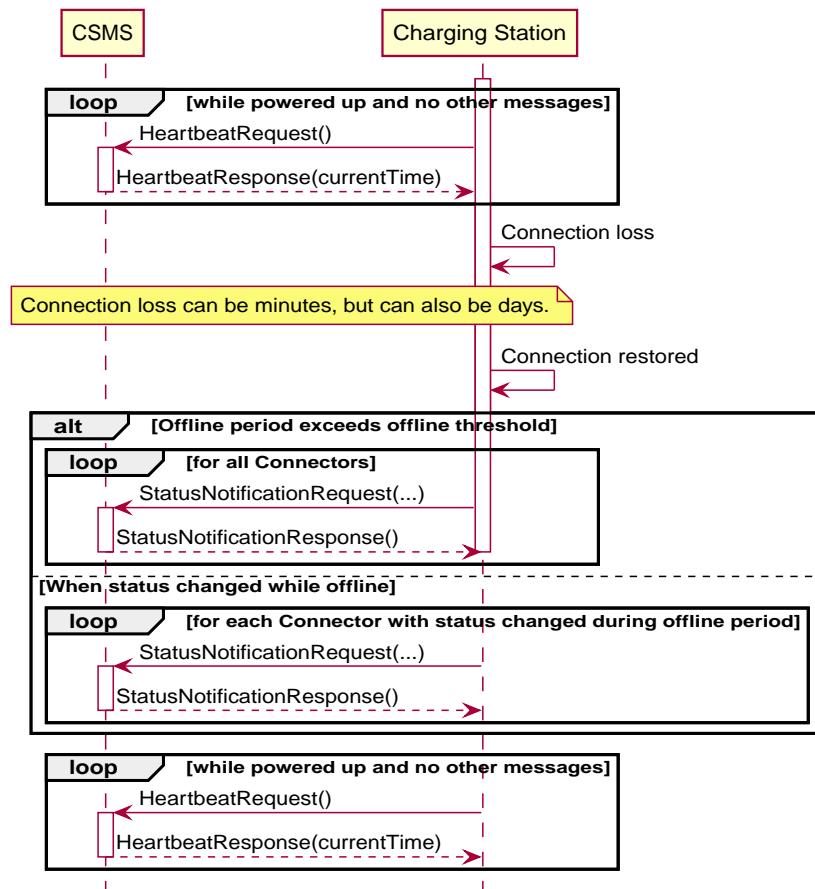


Figure 13. Sequence Diagram: Offline Behavior Idle Charging Station

7	Error handling	The offline situation is an non preferred mode of operation that needs to be handled by the Charging Station by trying to re-establish the connection.
8	Remark(s)	n/a

B04 - Offline Behavior Idle Charging Station - Requirements

Table 42. B04 - Requirements

ID	Precondition	Requirement definition
B04.FR.01	After having been <i>Offline</i> AND the <i>Offline</i> period exceeds the value of the <i>OfflineThreshold</i> Configuration Variable.	The Charging Station SHALL send StatusNotificationRequest to report the current status of all its Connectors.
B04.FR.02	After having been <i>Offline</i> AND the <i>Offline</i> period does NOT exceed the value of the <i>OfflineThreshold</i> Configuration Variable.	The Charging Station SHALL send StatusNotificationRequest to report the current status of only the Connectors for which a state change occurred.

2.2. Configuring a Charging Station

NOTE For managing the configuration of a Charging Station a basic understanding of Device Model concepts is essential. These concepts are explained in "OCPP 2.0.1: Part 1 - Architecture & Topology", chapter 4.

B05 - Set Variables

Table 43. B05 - Set Variables

No.	Type	Description
1	Name	Set Variables
2	ID	B05
	<i>Functional block</i>	B. Provisioning
3	Objective(s)	To give the CSMS the ability to make changes to variables in the Charging Station.
4	Description	A Charging Station can have a lot of variables that can be configured/changed by the CSMS. A CSMS can use these variables to for example influence the behavior of a Charging Station. This use case describes how the CSMS requests a Charging Station to set the value of variables of a component. The CSMS can request to set more than one value per request.
	Actors	CSMS, Charging Station
	<i>Scenario description</i>	<ol style="list-style-type: none"> 1. The CSO triggers the CSMS to request setting one or more variables in a Charging Station. 2. The CSMS sends a SetVariablesRequest to the Charging Station. 3. The Charging Station responds with a SetVariablesResponse indicating whether it was able to execute the change(s).
5	Prerequisite(s)	n/a
6	Postcondition(s)	Successful postconditions: <ol style="list-style-type: none"> 1. The change was executed <i>Successfully</i>. Failure postconditions: <ol style="list-style-type: none"> 1. The variable is supported, but setting could not be changed, the Charging Station responds with the status <i>Rejected</i>. 2. The variable is <i>not</i> supported, the Charging Station responds with the status <i>UnknownVariable</i>.

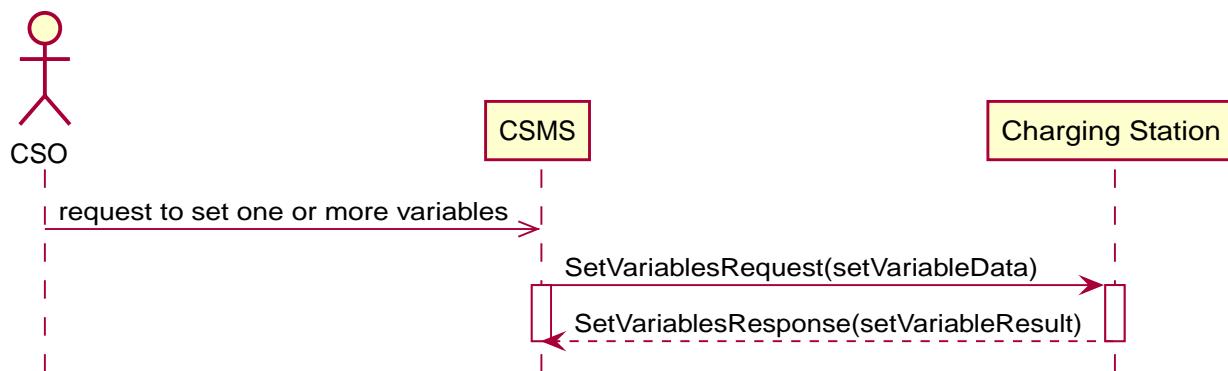


Figure 14. Sequence Diagram: Set Variables

7	Error handling	n/a
---	----------------	-----

8	Remark(s)	The attributeType Actual corresponds with the actual value of the Variable, whereas the attributeTypes Target, MinSet and MaxSet correspond to the target, minimum and maximum values that have been set for this variable. This is best explained by an example: the cooling system is configured to operate with a fan speed between 1000 and 5000 rpm. These boundaries are represented by the MinSet and MaxSet attributes. The current fan speed is represented by the Actual attribute. The desired fan speed is represented by the Target attribute.
---	------------------	--

B05 - Set Variables - Requirements

Table 44. B05 - Requirements

ID	Precondition	Requirement definition
B05.FR.01	When the Charging Station receives a SetVariablesRequest with an X number of SetVariableData elements	The Charging Station SHALL respond with an SetVariablesResponse with an equal (X) number of SetVariableResult elements, one for every SetVariableData element in the SetVariablesRequest .
B05.FR.02	B05.FR.01	Every SetVariableResult element in the SetVariablesResponse SHALL contain the same <i>component</i> and <i>variable</i> combination as one of the SetVariableData elements in the SetVariablesRequest .
B05.FR.03	B05.FR.02 AND If the SetVariablesRequest contains an <i>attributeType</i>	The corresponding SetVariableResult element in the SetVariablesResponse SHALL also contain the same <i>attributeType</i>
B05.FR.04	When the Charging Station receives a SetVariablesRequest with an unknown <i>Component</i> in the SetVariableData	The Charging Station SHALL set the <i>attributeStatus</i> field in the corresponding SetVariableResult to: UnknownComponent .
B05.FR.05	When the Charging Station receives a SetVariablesRequest with a <i>Variable</i> that is unknown for the given <i>Component</i> in the SetVariableData	The Charging Station SHALL set the <i>attributeStatus</i> field in the corresponding SetVariableResult to: UnknownVariable .
B05.FR.06	When the Charging Station receives a SetVariablesRequest with an <i>attributeType</i> that is unknown for the given <i>Variable</i> in the SetVariableData	The Charging Station SHALL set the <i>attributeStatus</i> field in the corresponding SetVariableResult to: NotSupportedAttributeType .
B05.FR.07	When the Charging Station receives a SetVariablesRequest with a <i>value</i> that is incorrectly formatted for the given <i>Variable</i> in the SetVariableData	The Charging Station SHALL set the <i>attributeStatus</i> field in the corresponding SetVariableResult to: Rejected . (More information can be provided in the optional <i>statusInfo</i> element.)
B05.FR.08	When the Charging Station receives a SetVariablesRequest with a <i>value</i> that is lower or higher than the range of the given <i>Variable</i> in the SetVariableData	The Charging Station SHALL set the <i>attributeStatus</i> field in the corresponding SetVariableResult to: Rejected . (More information can be provided in the optional <i>statusInfo</i> element.)
B05.FR.09	NOT (B05.FR.04 to B05.FR.08) AND When the Charging Station receives a SetVariablesRequest for a <i>Variable</i> in the SetVariableData , but is not able to set it	The Charging Station SHALL set the <i>attributeStatus</i> field in the corresponding SetVariableResult to: Rejected . (This happens if the variable is <i>ReadOnly</i> , but may also occur when setting the variable fails because of technical problems.)
B05.FR.10	When the Charging Station was able to set the given <i>value</i> from the SetVariableData	The Charging Station SHALL set the <i>attributeStatus</i> field in the corresponding SetVariableResult to: Accepted .
B05.FR.11		The CSMS SHALL NOT send more SetVariableData elements in a SetVariablesRequest than reported by the Charging Station via ItemsPerMessageSetVariables .
B05.FR.12	When the Charging Station receives a SetVariablesRequest without an <i>attributeType</i> .	The corresponding SetVariableResult element in the SetVariablesResponse SHALL contain the <i>attributeType</i> Actual .
B05.FR.13		The CSMS SHALL NOT include multiple SetVariableData elements, in a single SetVariablesRequest , with the same <i>Component</i> , <i>Variable</i> and <i>AttributeType</i> combination. Note that an omitted <i>AttributeType</i> counts as the value Actual .

B06 - Get Variables

Table 45. B06 - Get Variables

No.	Type	Description
1	Name	Get Variables
2	ID	B06
	Functional block	B. Provisioning
3	Objective(s)	To give the CSMS the ability to retrieve the value of an attribute for one or more Variables of one or more Components.
4	Description	This use case describes how the CSMS requests a Charging Station to send the value of an attribute for one or more variables of one or more components. It is not possible to get all attributes of all variables in one call.
	Actors	Charging Station, CSMS
	Scenario description	<ol style="list-style-type: none"> 1. The CSO triggers the CSMS to request for a number of variables in a Charging Station. 2. The CSMS request the Charging Station for a number of variables with GetVariablesRequest with a list of requested variables. 3. The Charging Station responds with a GetVariablesResponse with the requested variables. 4. The CSMS sends an optional notification to the CSO.
5	Prerequisite(s)	n/a
6	Postcondition(s)	<p>Successful postcondition: The Charging Station was able to send all the requested variables.</p> <p>Failure postcondition: The Charging Station was not able to send all requested variables.</p>

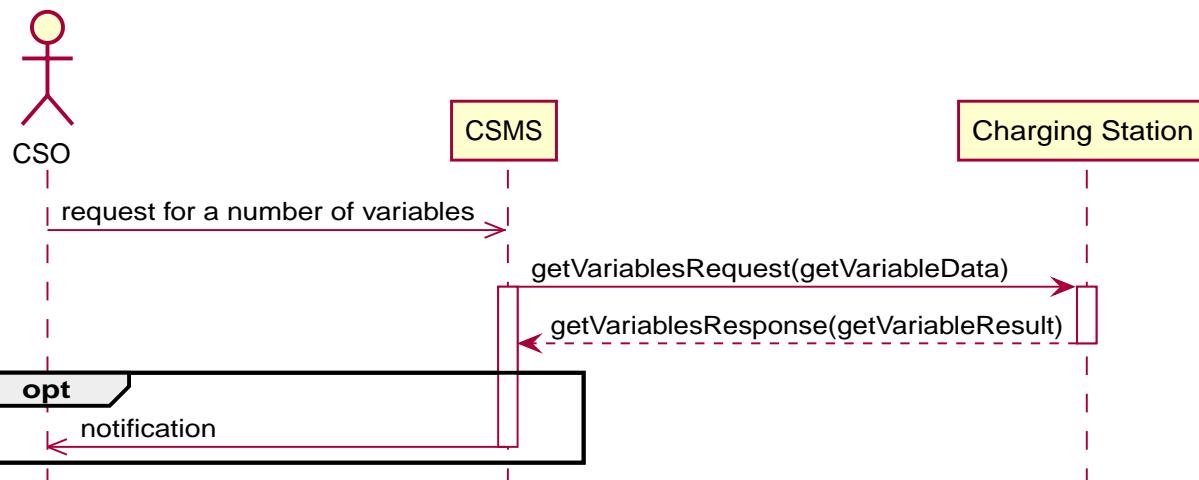


Figure 15. Sequence Diagram: Get Variables

7	Error handling	n/a
8	Remark(s)	n/a

B06 - Get Variables - Requirements

Table 46. B06 - Requirements

ID	Precondition	Requirement definition
B06.FR.01	When the Charging Station receives a GetVariablesRequest with an X number of GetVariableData elements	The Charging Station SHALL respond with an GetVariablesResponse with an equal (X) number of GetVariableResult elements, one for every GetVariableData element in the GetVariablesRequest .
B06.FR.02	B06.FR.01	Every GetVariableResult element in the GetVariablesResponse SHALL contain the same <i>component</i> and <i>variable</i> combination as one of the GetVariableData elements in the GetVariablesRequest .

ID	Precondition	Requirement definition
B06.FR.03	B06.FR.02 AND If the GetVariablesRequest contains an <i>attributeType</i>	The corresponding GetVariableResult element in the GetVariablesResponse SHALL also contain the same <i>attributeType</i>
B06.FR.04	B06.FR.01	Every GetVariableResult element in the GetVariablesResponse SHALL contain an <i>attributeValue</i> with the value of an attribute from the requested <i>attributeType</i> in the GetVariablesRequest .
B06.FR.05		The CSMS SHALL NOT send more GetVariableData elements in a GetVariablesRequest than reported by the Charging Station via ItemsPerMessageGetVariables .
B06.FR.06	When the Charging Station receives a GetVariablesRequest with an unknown Component in the GetVariableData	The Charging Station SHALL set the <i>attributeStatus</i> field in the corresponding GetVariableResult to: UnknownComponent AND SHALL omit the <i>attributeValue</i> .
B06.FR.07	When the Charging Station receives a GetVariablesRequest with a Variable that is unknown for the given Component in the GetVariableData	The Charging Station SHALL set the <i>attributeStatus</i> field in the corresponding GetVariableResult to: UnknownVariable AND SHALL omit the <i>attributeValue</i> .
B06.FR.08	When the Charging Station receives a GetVariablesRequest with an <i>attributeType</i> that is unknown for the given Variable in the GetVariableData	The Charging Station SHALL set the <i>attributeStatus</i> field in the corresponding GetVariableResult to: NotSupportedAttributeType AND SHALL omit the <i>attributeValue</i> .
B06.FR.09	When the Charging Station receives a GetVariablesRequest for a Variable in the GetVariableData that is <i>WriteOnly</i>	The Charging Station SHALL set the <i>attributeStatus</i> field in the corresponding GetVariableResult to: Rejected .
B06.FR.10	When the Charging Station was able to get the <i>value</i> requested from a GetVariablesRequest	The Charging Station SHALL set the <i>attributeStatus</i> field in the corresponding GetVariableResult to: Accepted and set the <i>attributeValue</i> to the found value.
B06.FR.11	When the Charging Station receives a GetVariablesRequest without an <i>attributeType</i> .	The corresponding GetVariableResult element in the GetVariablesResponse SHALL contain the <i>attributeType Actual</i> .
B06.FR.13	NOT B06.FR.08 AND the Charging Station has no <i>attributeValue</i> for the requested <i>attributeType</i> of the componentvariable	Charging Station SHALL return an empty string as <i>attributeValue</i> . Note: this can happen, for example, when the <i>attributeType Target</i> has not yet been set, even though it is supported.
B06.FR.14	B06.FR.01 AND a value for <i>instance</i> is provided in the <i>component</i> and/or <i>variable</i> in GetVariableData	Charging Station SHALL return the specified instance of that component and/or variable in GetVariableResult .
B06.FR.15	B06.FR.01 AND no value or an empty string is provided for <i>instance</i> in the <i>component</i> and/or <i>variable</i> in GetVariableData AND a component and/or variable without an <i>instance</i> does not exist	Charging Station SHALL return the <i>attributeStatus UnknownComponent</i> or <i>UnknownVariable</i> in the GetVariableResult entry for GetVariableData .
B06.FR.16	Charging Station receives a GetVariablesRequest with more GetVariableData elements than allowed by ItemsPerMessageGetVariables	The Charging Station MAY respond with a CALLERROR(<i>OccurrenceConstraintViolation</i>)
B06.FR.17	Charging Station receives a GetVariablesRequest with a length of more bytes than allowed by BytesPerMessageGetVariables	The Charging Station MAY respond with a CALLERROR(<i>FormatViolation</i>)

B07 - Get Base Report

Table 47. B07 - Get Base Report

No.	Type	Description
1	Name	Get Base Report
2	ID	B07
	Functional block	B. Provisioning
3	Objective(s)	To give the CSMS the ability to request a predefined report as defined in ReportBase .
4	Description	This use case describes how the CSMS requests a Charging Station to send a predefined report as defined in ReportBase . The result will be returned asynchronously in one or more NotifyReportRequest messages.
	Actors	Charging Station, CSMS
	Scenario description	<ol style="list-style-type: none"> 1. The CSO triggers the CSMS to request a report from a Charging Station. 2. The CSMS requests the Charging Station for a report with GetBaseReportRequest. 3. The Charging Station responds with GetBaseReportResponse. 4. The Charging Station asynchronously sends the results in one or more NotifyReportRequest messages. 5. The CSMS responds with NotifyReportResponse for each NotifyReportRequest.
5	Prerequisite(s)	n/a
6	Postcondition(s)	<p>Successful postcondition: The Charging Station was able to send the requested report.</p> <p>Failure postcondition: The Charging Station was not able to send the requested report.</p>

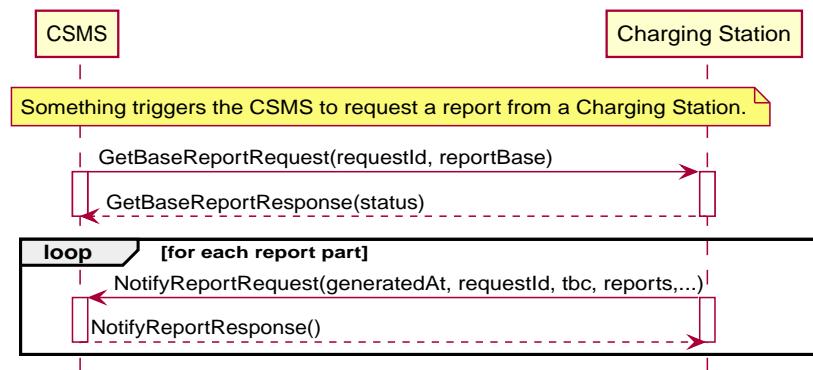


Figure 16. Sequence Diagram: Get Base Report

7	Error handling	n/a
8	Remark(s)	n/a

B07 - Get Base Report - Requirements

Table 48. B07 - Requirements

ID	Precondition	Requirement definition	Note
B07.FR.01	When the Charging Station receives a <code>getBaseReportRequest</code> for a supported <code>reportBase</code> AND NOT B07.FR.13	The Charging Station SHALL send a <code>getBaseReportResponse</code> with Accepted.	
B07.FR.02	When the Charging Station receives a <code>getBaseReportRequest</code> for a <code>reportBase</code> that is not supported	The Charging Station SHALL send a <code>getBaseReportResponse</code> with NotSupported.	

ID	Precondition	Requirement definition	Note
B07.FR.03	B07.FR.01	The Charging Station SHALL send the requested information via one or more NotifyReportRequest messages to the CSMS.	
B07.FR.04	B07.FR.01 AND The getBaseReportRequest contained a <i>requestId</i>	Every NotifyReportRequest send for this getBaseReportRequest SHALL contain the same <i>requestId</i> .	
B07.FR.05	B07.FR.02	The Charging Station SHALL NOT send a NotifyReportRequest to the CSMS.	
B07.FR.07	B07.FR.01 AND When <i>reportBase</i> is ConfigurationInventory	Then the Charging Station SHALL respond with a NotifyReportRequest to report on all component-variables that can be set by the operator including their <i>VariableCharacteristics</i> .	
B07.FR.08	B07.FR.01 AND When <i>reportBase</i> is FullInventory	Then the Charging Station SHALL respond with a NotifyReportRequest to report on all component-variables including their <i>VariableCharacteristics</i> .	As a minimum the required variables mentioned in Charging Infrastructure related shall be reported as well as the required variables in Section 1 Controller Components that are relevant to each functional block that has been implemented.
B07.FR.09	B07.FR.01 AND When <i>reportBase</i> is SummaryInventory	Then the Charging Station SHALL respond with a NotifyReportRequest to report on components and variables related to the availability and condition of the Charging Station, notably <i>operationalStatus</i> of the Charging Station, EVSE and Connectors and any error condition.	A (summary) report that lists Components/Variables relating to the Charging Station's current charging availability, and to any existing problem conditions. For the Charging Station Component: - <i>AvailabilityState</i> . For each EVSE Component: - <i>AvailabilityState</i> . For each Connector Component: - <i>AvailabilityState</i> (if known and different from EVSE). For all Components in an abnormal State: - Active (Problem, Tripped, Overload, Fallback) variables. - Any other diagnostically relevant Variables of the Components.
B07.FR.10		The sequence number contained in the <i>seqNo</i> field of the NotifyReportRequest is incremental per report. So the NotifyReportRequest message which contains the first report part, SHALL have a <i>seqNo</i> with value 0.	
B07.FR.11	B07.FR.08	All attribute types of a variable, that are supported by the Charging Station, SHALL be reported, even if they have no value (are unset).	This allows a CSMS to know which attribute types are supported by the Charging Station.
B07.FR.12		The Charging Station SHALL support at least the base reports: ConfigurationInventory and FullInventory .	
B07.FR.13	When the Charging Station is temporarily unable to execute a report request	The Charging Station SHALL send a getBaseReportResponse with Rejected.	

ID	Precondition	Requirement definition	Note
B07.FR.14	When a Charging Station connects to CSMS for the first time OR whenever CSMS suspects that the device model of the Charging Station has changed (e.g. after a firmware update or hardware change)	CSMS SHOULD request a GetBaseReportRequest with <i>reportBase</i> = <code>FullInventory</code> to retrieve a complete list of all its device model components and variables.	It is not mandated, because implementations may exist that are based on a known set of charging stations with fixed device models that will not change.

B08 - Get Custom Report

Table 49. B08 - Get Custom Report

No.	Type	Description
1	Name	Get Custom Report
2	ID	B08
	Functional block	B. Provisioning
3	Objective(s)	To give the CSMS the ability to request a report of all Components and Variables limited to those that match ComponentCriteria and/or the list of ComponentVariables.
4	Description	This use case describes how the CSMS requests a Charging Station to send a report of all Components and Variables limited to those that match ComponentCriteria and/or the list of ComponentVariables. The result will be returned asynchronously in one or more NotifyReportRequest messages.
	Actors	Charging Station, CSMS
	Scenario description	<ol style="list-style-type: none"> 1. The CSO triggers the CSMS to request a report from a Charging Station. 2. The CSMS requests the Charging Station for a report with a GetReportRequest. 3. The Charging Station responds with a GetReportResponse. 4. The Charging Station asynchronously sends the results in one or more NotifyReportRequest messages. 5. The CSMS responds with a NotifyReportResponse.
5	Prerequisite(s)	n/a
6	Postcondition(s)	<p>Successful postcondition: The Charging Station was able to send the requested report.</p> <p>Failure postcondition: The Charging Station was not able to send the requested report.</p>

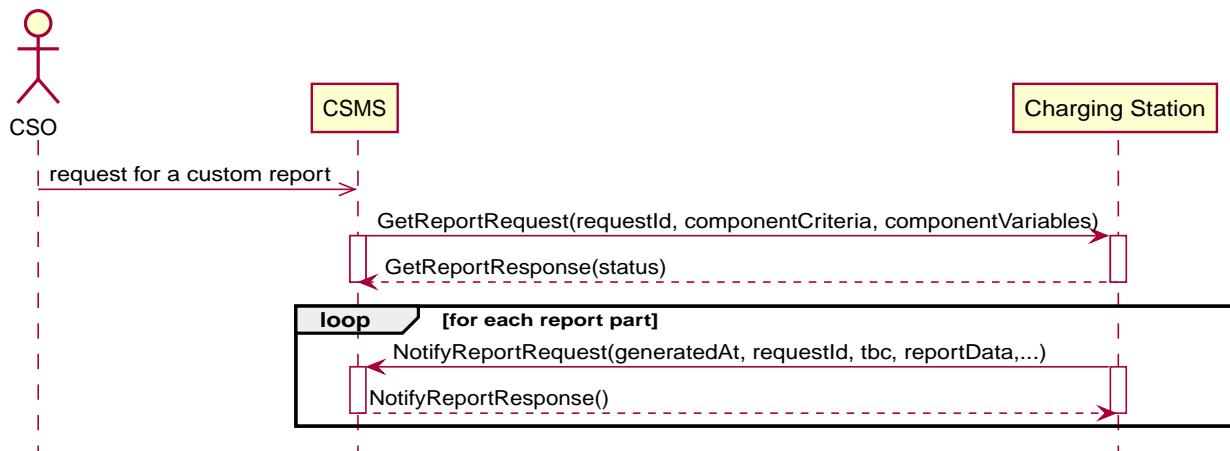


Figure 17. Sequence Diagram: Get Custom Report

7	Error handling	n/a
8	Remark(s)	n/a

B08 - Get Custom Report - Requirements

Table 50. B08 - Requirements

ID	Precondition	Requirement definition
B08.FR.01	NOT B08.FR.15 AND When the Charging Station receives a getReportRequest for supported criteria	The Charging Station SHALL send a getReportResponse with Accepted
B08.FR.02	When the Charging Station receives a getReportRequest for not supported criteria	The Charging Station SHALL send a getReportResponse with NotSupported
B08.FR.03	B08.FR.01	The Charging Station SHALL send the requested information via one or more NotifyReportRequest messages to the CSMS.

ID	Precondition	Requirement definition
B08.FR.04	B08.FR.01 AND The GetReportRequest contained a <i>requestId</i>	Every NotifyReportRequest sent for this getReportRequest SHALL contain the same <i>requestId</i> .
B08.FR.05	B08.FR.01 AND <i>componentCriteria</i> and <i>componentVariables</i> are NOT both empty.	Every NotifyReportRequest sent for this getReportRequest SHALL be limited to the set <i>componentCriteria</i> and <i>componentVariables</i> .
B08.FR.06		The maximum number of <i>componentVariables</i> in one getReportRequest message is given by the ItemsPerMessageGetReport Configuration Variable
B08.FR.07	B08.FR.01 AND <i>ComponentCriteria</i> contains: <i>Active</i>	The Charging Station SHALL report every component that has the variable <i>Active</i> set to <i>true</i> , or does not have the <i>Active</i> variable in a NotifyReportRequest .
B08.FR.08	B08.FR.01 AND <i>ComponentCriteria</i> contains: <i>Available</i>	The Charging Station SHALL report every component that has the variable <i>Available</i> set to <i>true</i> , or does not have the <i>Available</i> variable, in a NotifyReportRequest .
B08.FR.09	B08.FR.01 AND <i>ComponentCriteria</i> contains: <i>Enabled</i>	The Charging Station SHALL report every component that has the variable <i>Enabled</i> set to <i>true</i> , or does not have the <i>Enabled</i> variable, in a NotifyReportRequest .
B08.FR.10	B08.FR.01 AND <i>ComponentCriteria</i> contains: <i>Problem</i>	The Charging Station SHALL report every component that has the variable <i>Problem</i> set to <i>true</i> in a NotifyReportRequest .
B08.FR.11	B08.FR.01 AND <i>componentCriteria</i> is absent AND <i>componentVariables</i> is NOT empty.	Every NotifyReportRequest sent for this getReportRequest is limited to the set in <i>componentVariables</i> .
B08.FR.12	B08.FR.01	The reported variables in NotifyReportRequest SHALL contain <i>variableCharacteristics</i> .
B08.FR.13	B08.FR.01 AND More than one <i>componentCriteria</i> is given.	The Charging Station SHALL report all components that have at least one of the given criteria (logical OR).
B08.FR.14		The sequence number contained in the <i>seqNo</i> field of the NotifyReportRequest is incremental per report. So the NotifyReportRequest message which contains the first report part, SHALL have a <i>seqNo</i> with value 0.
B08.FR.15	When the Charging Station receives a GetReportRequest with a combination of criteria which results in an empty result set.	The Charging Station SHALL respond with a GetReportResponse(status=EmptyResultSet) .
B08.FR.16	When the Charging Station is temporarily unable to execute a report request	The Charging Station SHALL send a getBaseReportResponse with <i>Rejected</i> .
B08.FR.17	Charging Station receives a GetReportRequest with more <i>ComponentVariableType</i> elements than allowed by ItemsPerMessageGetReport	The Charging Station MAY respond with a CALLERROR(OccurrenceConstraintViolation)
B08.FR.18	Charging Station receives a GetReportRequest with a length of more bytes than allowed by BytesPerMessageGetReport	The Charging Station MAY respond with a CALLERROR(FormatViolation)
B08.FR.19	When Charging Station receives a GetReportRequest with <i>componentVariable</i> elements in which <i>component.instance</i> and/or <i>component.evse</i> are missing	The Charging Station SHALL report for every instance and/or EVSE of the <i>component</i> in <i>componentVariable</i> .
B08.FR.20	When Charging Station receives a GetReportRequest with <i>componentVariable</i> elements in which <i>variable</i> is missing	The Charging Station SHALL report for every <i>variable</i> of the <i>component</i> in <i>componentVariable</i> .
B08.FR.21	When Charging Station receives a GetReportRequest with <i>componentVariable</i> elements in which <i>variable</i> is present, but <i>instance</i> is missing	The Charging Station SHALL report for every instance of the <i>variable</i> of the <i>component</i> in <i>componentVariable</i> .

B09 - Setting a new NetworkConnectionProfile

Table 51. B09 - Setting a new NetworkConnectionProfile

No.	Type	Description
1	Name	Setting a new NetworkConnectionProfile.
2	ID	B09
	Functional block	B. Provisioning
3	Objectives	To enable the CSMS to update the connection details on the Charging Station.
4	Description	The CSMS updates the connection details on the Charging Station. For instance in preparation of a migration to a new CSMS. After completion of this use case, the Charging Station to CSMS connection data has been updated.
	Actors	Charging Station, CSMS
	Scenario description	<ol style="list-style-type: none"> 1. The CSMS sends a SetNetworkProfileRequest PDU containing an updated connection profile 2. The Charging Station receives the PDU, validates the content and stores the new data 3. The Charging Station responds by sending a SetNetworkProfileResponse PDU, with status Accepted
5	Prerequisites	The data supplied by the CSMS matches the Charging Station's capabilities
6	Postcondition(s)	The Charging Station was able to store the new connection data

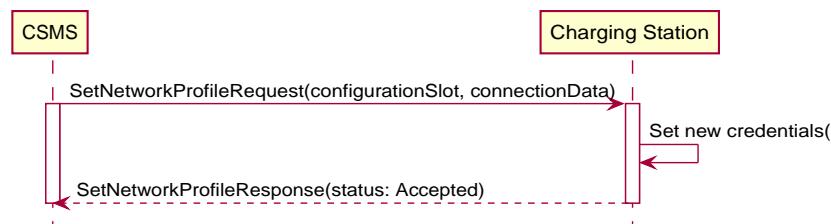


Figure 18. Sequence Diagram: Set Network Connection Profile

8	Error Handling	Activation of a new NetworkConnectionProfile is described in B10 - Migrate to new CSMS . Errors during this use-case are not destructive to the current data connection. Error handling is further described in B10 - Migrate to new CSMS
9	Remarks	Even when changes are made to the currently active NetworkConnectionProfile, these will not be activated until a reboot has occurred, as described in B10 - Migrate to new CSMS .

B09 - Setting a new NetworkConnectionProfile - Requirements

Table 52. B09 - Requirements

ID	Precondition	Requirement definition
B09.FR.01	On receipt of the SetNetworkProfileRequest	The Charging Station SHALL validate the content, store the new data and if successful, respond by sending a SetNetworkProfileResponse message, with status <code>Accepted</code>
B09.FR.02	On receipt of the SetNetworkProfileRequest	The Charging Station SHALL validate the content. If the content is invalid, the Charging Station SHALL respond by sending a SetNetworkProfileResponse message, with status <code>Rejected</code>
B09.FR.03	If setting the new networkprofile fails.	The Charging Station SHALL respond by sending a SetNetworkProfileResponse message, with status <code>Failed</code>
B09.FR.04	On receipt of the SetNetworkProfileRequest AND the NetworkConnectionProfile contains a lower securityProfile than stored at the configuration variable SecurityProfile	The Charging Station SHALL respond by sending a SetNetworkProfileResponse message, with status <code>Rejected</code>

B10 - Migrate to new CSMS

Table 53. B10 - Migrate to new CSMS

No.	Type	Description
1	Name	Migrate to new CSMS, using a different NetworkConnectionProfile.
2	ID	B10
	<i>Functional block</i>	B. Provisioning
3	Objectives	After completion of this use case, the Charging Station connects to a new CSMS.
4	Description	This use case describes how a Charging Station can be instructed to connect to a new CSMS, by changing the order of NetworkConnectionProfiles in NetworkConfigurationPriority .
	Actors	Charging Station, CSMS 1, CSMS 2
	Scenario description	<p>1. CSMS 1 sets a new value for the NetworkConfigurationPriority Configuration Variable via SetVariablesRequest, such that the NetworkConnectionProfile for CSMS 2 becomes first in the list and the existing connection to CSMS 1 becomes second in the list.</p> <p>2. The Charging Station responds with a SetVariablesResponse with status Accepted</p> <p>3. CSMS 1 instructs the Charging Station to perform a Reset OnIdle.</p> <p>4. The Charging Station reboots and connects via the new primary NetworkConnectionProfile to CSMS 2.</p>
5	Prerequisites	<p>Use case B09 - Setting a new NetworkConnectionProfile was executed successfully prior to this use case</p> <p>The data supplied by the CSMS matches the Charging Station's capabilities</p>
6	Postcondition(s)	The Charging Station is connected via a different NetworkConnectionProfile .

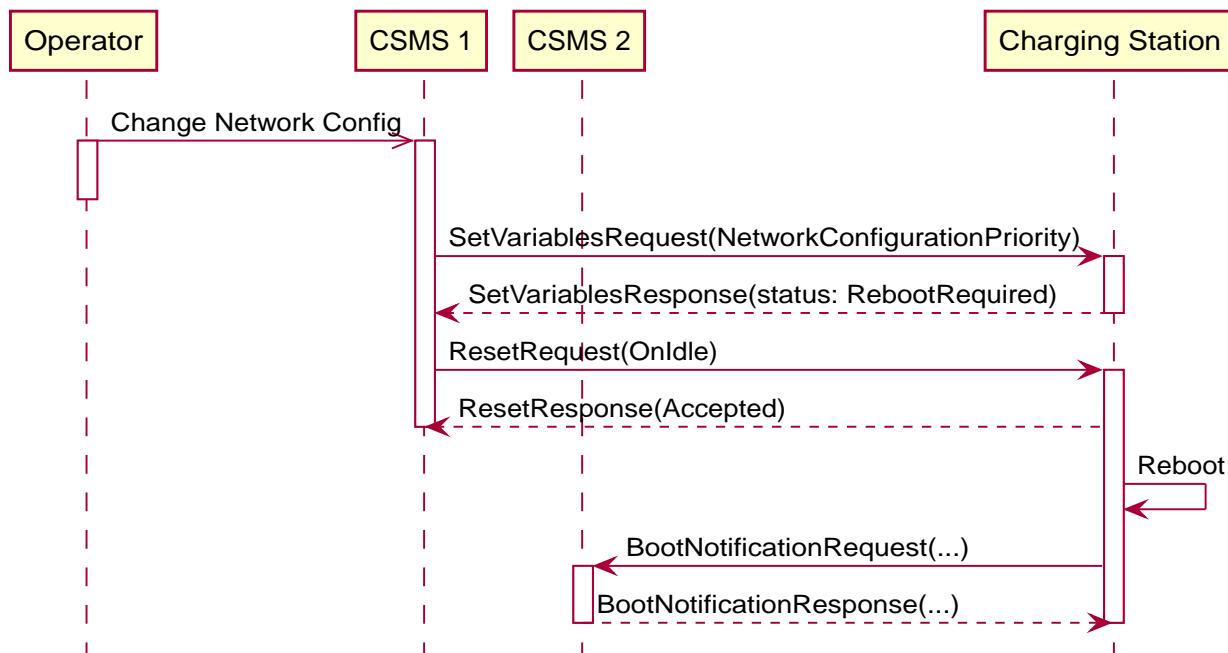


Figure 19. Sequence Diagram: Migrate to new ConnectionProfile

8	Error Handling	n/a
9	Remarks	<p>As in line with B12 - Reset - With Ongoing Transaction, when there are ongoing transactions, the Charging Station waits for these to be finished before performing the Reset and then connecting to a different CSMS.</p> <p>When an operator wants to perform an immediate switch, he should stop the transactions first.</p>

B10 - Migrate to new NetworkConnectionProfile - Requirements

Table 54. B10 - Requirements

ID	Precondition	Requirement definition	Note
B10.FR.01	On receipt of a SetVariablesRequest , containing Configuration Variable NetworkConfigurationPriority AND the NetworkProfile slots in the message all contain valid configurations	The Charging Station SHALL send SetVariablesResponse with status Accepted, or RebootRequired .	
B10.FR.02	On receipt of a SetVariablesRequest , containing Configuration Variable NetworkConfigurationPriority AND any of the NetworkProfile slots in the message does not contain a valid configuration	The Charging Station SHALL send SetVariablesResponse with status Rejected.	The optional element statusInfo can be used to provide more information.
B10.FR.03	B10.FR.04 AND When connecting fails	The Charging Station SHALL make the number of attempts as configured in NetworkProfileConnectionAttempts per entry of NetworkConfigurationPriority .	
B10.FR.04	B10.FR.01 OR B09.FR.01 AND After a reboot	The Charging Station SHALL begin connecting to the first entry of NetworkConfigurationPriority	
B10.FR.05		It is RECOMMENDED to set the Charging Station to Inoperative (via ChangeAvailabilityRequest) to ensure that no new transactions can be started and wait until the transaction message queue in the Charging Station is empty before sending the ResetRequest . Otherwise the Charging Station might send transaction related messages to the new CSMS that has not received the start of the Transaction, and the old system will miss the ended messages. To determine if there are still transaction for an ongoing transaction in the queue, the getTransactionStatusRequest message can be used.	
B10.FR.06		The Charging Station SHALL disconnect from the old CSMS, before trying to connect to the new CSMS.	
B10.FR.07	B10.FR.03 AND All NetworkProfileConnectionAttempts for every entry of NetworkConfigurationPriority failed.	The Charging Station SHOULD fallback and start 'reconnecting' to the NetworkConnectionProfile for which the last successful connection was made.	'reconnecting' in this requirement, refers to the reconnection mechanism described at section 5.3. Reconnecting from "Part 4 - JSON over WebSockets implementation guide".

2.3. Resetting a Charging Station

B11 - Reset - Without Ongoing Transaction

Table 55. B11 - Reset - Without Ongoing Transaction

No.	Type	Description
1	Name	Reset - Without Ongoing Transaction
2	ID	B11
	<i>Functional block</i>	B. Provisioning
3	Objective(s)	To enable the CSMS to request a Charging Station to reset itself or an EVSE, while there is no ongoing transaction.
4	Description	This use case covers how the CSMS can request the Charging Station to reset itself or an EVSE by sending ResetRequest . (If ResetRequest contains an optional parameter evselid , then only a reset of the specific EVSE is requested.) This could for example be necessary if the Charging Station is not functioning correctly.
	Actors	Charging Station, CSMS, CSO

No.	Type	Description
	Scenario description	<p>1. The CSO requests the CSMS to reset the Charging Station or EVSE.</p> <p>2. The CSMS sends ResetRequest requesting the Charging Station to reset itself or EVSE.</p> <p>3. The CSMS requests for an OnIdle or Immediate reset.</p> <p>4. The Charging Station responds with ResetResponse, indicating whether the Charging Station is able to reset itself or EVSE.</p> <p>5. The CSMS sends an optional notification to the CSO.</p> <p>6. Only if no evseld was supplied, then after the reset, the Charging Station will proceed as in use case B01.</p>
	Alternative scenario(s)	B12 - Reset With Ongoing Transaction
5	Prerequisite(s)	No transaction is ongoing.
6	Postcondition(s)	<p>Successful postcondition: The Charging Station was able to reset itself or EVSE.</p> <p>Failure postcondition: The Charging Station <i>not</i> was able to reset itself or EVSE.</p>

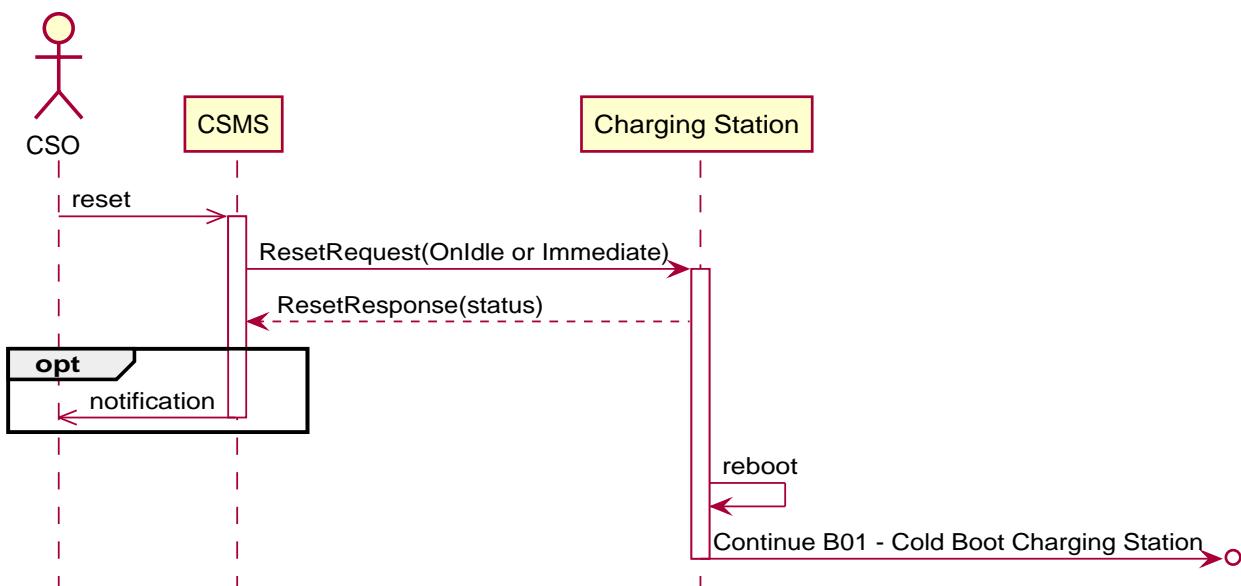


Figure 20. Sequence Diagram: Reset Without Transaction

7	Error handling	n.a
8	Remark(s)	<p>Persistent states: for example, EVSE set to <i>Unavailable</i> SHALL persist.</p> <p>The Charging Station responds with ResetResponse.</p>

B11 - Reset - Without Ongoing Transaction - Requirements

Table 56. B11 - Requirements

ID	Precondition	Requirement definition
B11.FR.01	When the Charging Station receives a ResetRequest .	The Charging Station SHALL respond with a ResetResponse .
B11.FR.02	If the status was set to <i>Inoperative</i> by the CSMS.	After a reboot of the Charging Station, the EVSEs SHALL return to the state <i>Unavailable</i> as prior to the reboot.
B11.FR.03	B11.FR.01 AND no evseld parameter is supplied AND ResetResponse was Accepted.	The Charging Station MAY send a StatusNotification(Unavailable) and SHALL start a reboot.
B11.FR.04	B11.FR.03	The Charging Station SHALL proceed as described in use case B01 - Cold Boot Charging Station .

ID	Precondition	Requirement definition
B11.FR.05	If the status of an EVSE was <i>Reserved</i> .	After a reboot of the Charging Station or EVSE, the EVSE(s) SHALL return to the state <i>Reserved</i> .
B11.FR.06	B11.FR.01 AND For example there is a firmware update ongoing that cannot be interrupted.	The Charging Station SHALL respond with a status <i>Rejected</i> .
B11.FR.07	B11.FR.01 AND Charging Station cannot perform the reset now, but has scheduled the reset for later	The Charging Station SHALL respond with a status <i>Scheduled</i> .
B11.FR.08	B11.FR.01 AND an <i>evselid</i> parameter is supplied AND ResetResponse was <i>Accepted</i> .	The Charging Station MAY send a StatusNotification(<i>Unavailable</i>) for the EVSE and SHALL start a reboot of EVSE that is referred to by <i>evselid</i> parameter.
B11.FR.09	B11.FR.01 AND an <i>evselid</i> parameter is supplied AND Charging Station does not support resetting an individual EVSE	The Charging Station SHALL return a ResetResponse <i>Rejected</i>
B11.FR.10	When the Charging Station supports resetting of an individual EVSE	The Charging Station SHOULD set the device model variable AllowReset to true for the EVSE.

B12 - Reset - With Ongoing Transaction

Table 57. B12 - Reset - With Ongoing Transaction

No.	Type	Description
1	Name	Reset - With Ongoing Transaction
2	ID	B12
	<i>Functional block</i>	B. Provisioning
3	Objective(s)	To enable the CSMS to request a Charging Station to reset itself or EVSE, while there is an ongoing transaction.
4	Description	This use case covers how the CSMS can request the Charging Station to reset itself or an EVSE by sending ResetRequest . (If ResetRequest contains an optional parameter <code>evseld</code> , then only a reset of the specific EVSE is requested.) This could for example be necessary if the Charging Station is not functioning correctly. The CSMS has the possibility to let the Charging Station end all transactions itself and reboot or wait until all ongoing transactions are ended normally (by an EV user) and then reboot.
	Actors	Charging Station, CSMS, CSO
	Scenario description	<p>1. The CSO requests the CSMS to reset the Charging Station or EVSE.</p> <p>2. The CSMS sends ResetRequest requesting the Charging Station to reset itself or EVSE.</p> <p>3a. On receipt of an Onidle reset, the Charging Station responds with ResetResponse(Scheduled), indicating the Charging Station will try to reset itself or EVSE after all ongoing transactions have ended. The Charging Station continues charging and sets all EVSEs (or only the one provided in the request, if <code>evseld</code> was supplied) that are Available to status <i>Unavailable</i>, waits until all transactions are finished and all TransactionEventRequest (<code>eventType = Ended</code>) messages are sent.</p> <p>3b. On receipt of an Immediate reset, the Charging Station responds with ResetResponse(Accepted), indicating the Charging Station will try to reset itself or EVSE. The Charging Station attempts to terminate any transaction (or only those running on the EVSE provided in the request, if <code>evseld</code> was supplied) in progress, and sending a TransactionEventRequest (<code>eventType = Ended</code>) message.</p> <p>4. Only if no <code>evseld</code> was supplied the Charging Station reboots and returns to a state as just having been booted, B01 - Cold Boot Charging Station applies.</p>
	Alternative scenario(s)	B11 - Reset Without Ongoing Transaction
5	Prerequisite(s)	A transaction is ongoing.
6	Postcondition(s)	<p>Successful postcondition: The Charging Station was able to reset itself or EVSE.</p> <p>Failure postcondition: The Charging Station <i>not</i> was able to reset itself or EVSE.</p>

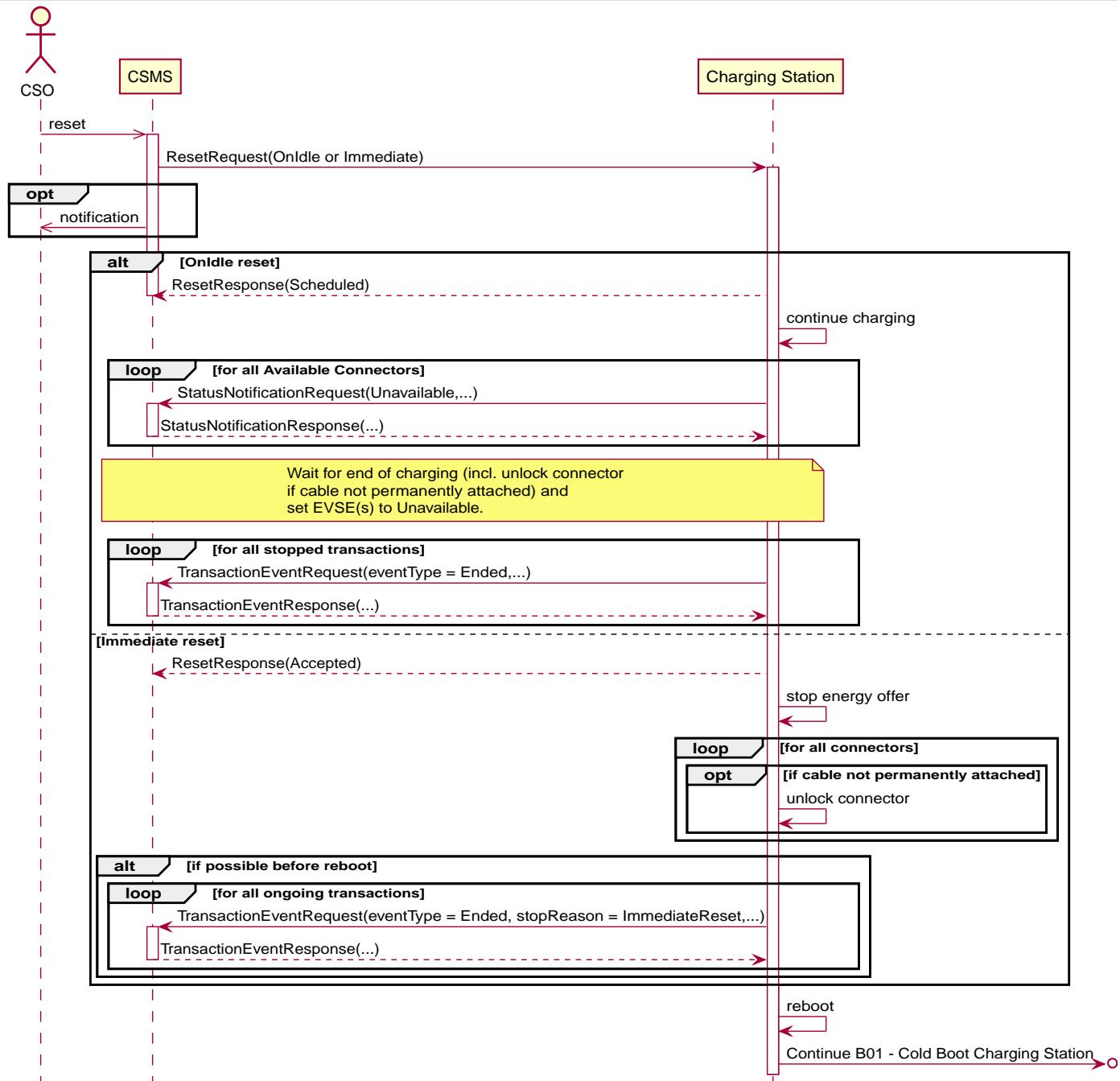


Figure 21. Sequence Diagram: Reset With Ongoing Transaction

7	Error handling	After having accepted the <code>ResetRequest</code> , <code>TransactionEventRequest</code> messages that cannot be delivered to the CSMS MUST be queued.
8	Remark(s)	n/a

B12 - Reset - With Ongoing Transaction - Requirements

Table 58. B12 - Requirements

ID	Precondition	Requirement definition
B12.FR.01	When the Charging Station receives a <code>ResetRequest(OnIdle)</code> AND a transaction is ongoing	The Charging Station SHALL respond with a <code>ResetResponse(Scheduled)</code> , to indicate whether the Charging Station will attempt to reset itself or EVSE after all transactions on Charging Station or EVSE have ended.
B12.FR.02	When the Charging Station receives a <code>ResetRequest(Immediate)</code> AND a transaction is ongoing	The Charging Station SHALL respond with a <code>ResetResponse(Accepted)</code> , to indicate whether the Charging Station will attempt to reset itself or EVSE.

ID	Precondition	Requirement definition
B12.FR.03	If no <i>evselid</i> is supplied AND If any transaction is in progress and an OnIdle reset is received.	The transaction of the Charging Station SHALL be terminated normally, before the reboot, as in E06 - Stop Transaction .
B12.FR.04	If no <i>evselid</i> is supplied AND If any transaction is in progress and an Immediate Reset is received.	The Charging Station SHALL attempt to terminate any transaction in progress and send a TransactionEventRequest (eventType = Ended) message before performing a reboot.
B12.FR.05	If an Immediate Reset is received and the TransactionEventResponse is not received within timeout.	The Charging Station SHALL queue the TransactionEventRequest , reboot and resend the TransactionEventRequest after the reboot.
B12.FR.06	If the status was set to <i>Inoperative</i> by the CSMS.	After a reboot of the Charging Station or EVSE, the EVSE(s) SHALL return to the state <i>Unavailable</i> as prior to the reboot.
B12.FR.07	If an <i>evselid</i> is supplied AND If a transaction is in progress on the EVSE and an OnIdle reset is received.	The transaction on the EVSE SHALL be terminated normally, before the reboot, as in E06 - Stop Transaction .
B12.FR.08	If an <i>evselid</i> is supplied AND If a transaction is in progress on the EVSE and an Immediate Reset is received.	The Charging Station SHALL attempt to terminate the transaction in progress on the EVSE and send a TransactionEventRequest (eventType = Ended) message before performing a reboot.
B12.FR.09	B12.FR.01 AND an <i>evselid</i> parameter is supplied AND Charging Station does not support resetting an individual EVSE	The Charging Station SHALL return a ResetResponse Rejected

C. Authorization

1. Introduction

This Functional Block describes all the authorization-related functionalities, it contains different ways of authorizing a user, online and/or offline and the AuthorizeRequest message handling/behavior, Authorization Cache functionality, etc.

When a user wishes to unplug the electric vehicle from the Charging Station, the Charging Station needs to verify that the user is either the one that initiated the charging or that the user is in the same group and thus allowed to terminate the charging. Once authorized, the Charging Station informs the CSMS that the charging has been stopped.

- To improve the experience for users, a Charging Station MAY support local authorization of identifiers, using an [Authorization Cache](#).
- The [LocalAuthorizeOffline](#) Configuration Variable controls whether a Charging Station will authorize a user when *offline* using the Authorization Cache.
- The [LocalPreAuthorize](#) Configuration Variable controls whether a Charging Station will use the Authorization Cache to start a transaction without performing an authorization with the CSMS.

1.1. ID Tokens

This section is normative

OCPP now makes it possible to use many different types of authorization. Where OCPP 1.x only supported RFID, OCPP now also supports things like: credit card, PIN-code, a simple start button etc.

An [IDTokenType](#) contains the identifier to use for authorization. It is defined as a combination of a case insensitive string and a type. Message data elements of the [IDTokenType](#) class (including GroupId) MAY contain any data, that is meaningful to a CSMS (e.g. for the purpose of identifying the initiator of charging activity), and Charging Stations MUST NOT make any presumptions as to the format or content of such data, other than is provided in the description of the [IDTokenType](#) (e.g. by assuming that it is a UID-like value that must be hex characters only and/or an even number of digits). IdToken data acquired via local token reader hardware is usually a (4, 7 or 10 bytes) UID value of a physical IdToken, typically represented as 8, 14 or 20 hexadecimal digit characters.

NOTE To promote interoperability, based on common practice to date in the case of [IdTokenType](#) data has type: [ISO14443](#), it is RECOMMENDED that such UIDs be represented as hex representations of the UID bytes. According to ISO 14443-3, byte 0 should come first in the hex string. (Most significant nibble of byte 0 first)

1.1.1. Additional Info

[AdditionalInfo](#) can be used to send extra information which can be validated by the CSMS in addition to the regular authorization with *IdToken*.

[AdditionalInfo](#) contains one or more custom types, which need to be agreed upon by all parties involved. When [AdditionalInfo](#) is implemented the Charging Station SHALL also cache and include [AdditionalInfo](#) during regular operations and set the Configuration Variable [AdditionalInfoItemsPerMessage](#). When [AdditionalInfo](#) is NOT implemented or a not supported [AdditionalInfo.type](#) is used, the CSMS/Charging Station MAY ignore the [AdditionalInfo](#).

1.2. Group ID Tokens

This section is normative

A CSMS has the ability to treat a set of identity tokens as a "group", thereby allowing any one token in the group to start a transaction and for the same token, or another token in the same group, to stop the transaction. This supports the common use-cases of families or businesses with multiple drivers using one or more shared electric vehicles on a single recharging contract account. [IDTokenType](#)s used as "GroupId" may often use a shared central account identifier for the GroupId, instead of a UID of the first/master RFID card of an account.

Tokens (*idTags*) are grouped for authorization purposes by specifying a common group identifier in the optional *groupIdToken* element in [IdTokenInfo](#): two IdTokens are considered to be in the same group if their *GroupIdTokens* match (and they are not empty).

NOTE Even though the GroupId has the same nominal data type ([IdTokenType](#)) as an *idToken*, the value of this element may not be in the common format of [IDTokenType](#)s and/or may not represent an actual valid [IdTokenType](#) (e.g. it may be a common shared "account number"): therefore, the GroupId value SHOULD NOT be used for comparison against a presented Token value (unless it also occurs as an *idToken* value).

1.3. Authorization Cache

A Charging Station MAY implement an Authorization Cache that **autonomously** maintains a record of previously presented identifiers that have been successfully authorized by the CSMS. The Authorization Cache can be used to speed up the authorization process at the Charging Station, since using a locally stored cache means that the user does not have to wait for the Charging Station to check the authorization at the CSMS. Operation of the Authorization Cache, when present, is reported (and controlled, where possible) by the [AuthCacheEnabled](#) Configuration Variable. The optional expiration time of general Authorization Cache entries can be set in the Configuration Variable [AuthCacheLifeTime](#). If a different expiration time is desired for a specific entry, this can be set in the `cacheExpiryDateTime` that is returned in `iDTokenInfo` of, for example, the [AuthorizeResponse](#).

Please refer to the use cases [C10 - Store Authorization Data in the Authorization Cache](#) and [C12 - Start Transaction - Cached Id](#) for more information on how to implement / use the Authorization Cache functionality.

When a Charging Station supports both the Authorization Cache and Tariff information (see: [Tariff & Cost](#)), it should not store the tariff information in the Authorization Cache, since this information could become outdated.

A Charging Station MAY support the authorization of *any* presented identifier when *offline*, to avoid refusal of charging to bona fide users that cannot be explicitly authorized by [Authorization Cache](#) entries. This functionality is explained in more detail in [Unknown Offline Authorization](#).

It is RECOMMENDED to store personal information in the Authorization Cache securely, e.g. by only storing hashed idTokens in the cache.

1.4. Local Authorization List

The Local Authorization List is a list of identifiers that can be synchronized with the CSMS. It allows authorization of a user when offline and faster (apparent) authorization response time when communication between Charging Station and CSMS is slow. The CSMS can synchronize the list by either sending a complete list of identifiers to replace the Local Authorization List or by sending a list of changes (add, update, delete) to apply to the Local Authorization List. The operations to support this are [GetLocalListVersion](#) and [SendLocalList](#).

This list contains the authorization status of all (or a selection of) identifiers and the corresponding expiration date. These values may be used to provide more fine grained information to users (e.g. by display message) during local authorization.

Please refer to the use cases [D01 - Send Local Authorization List](#), [C13 - Offline Authorization through Local Authorization List](#) and [C14 - Online Authorization through Local Authorization List](#) for more information on how to implement / use the Local Authorization List functionality.

NOTE

Please note the difference between the [Authorization Cache](#) and [Local Authorization List](#) mechanisms: the [Authorization Cache](#) is an autonomous mechanism at the Charging Station, whereas the [Local Authorization List](#) is a list that is synchronized between CSMS and Charging Station (originating from the CSMS).

NOTE

The [Authorization Cache](#) and [Local Authorization List](#) are **distinct** logical data structures. When both [Authorization Cache](#) as well as [Local Authorization List](#) are supported, a Charging Station SHALL treat [Local Authorization List](#) entries as having priority over [Authorization Cache](#) entries for the same identifiers.

The following Configuration Variables are used by the Charging Station to give information about the Local Authorization List

- [LocalAuthListEntries](#) (Also reports the maximum amount of IdTokens in the Local Authorization List)
- [LocalAuthListEnabled](#)
- [LocalAuthListAvailable](#)
- [ItemsPerMessageSendLocalList](#)
- [BytesPerMessageSendLocalList](#)

1.5. Unknown Offline Authorization

When *offline*, a Charging Station MAY allow automatic authorization of any "unknown" identifiers that are not found in the [Local Authorization List](#) and/or [Authorization Cache](#). Operation of the Unknown Offline Authorization capability, when supported, is reported (and controlled, where possible) by the [OfflineTxForUnknownIdEnabled](#) Configuration Variable. When connection to the CSMS is restored, the Charging Station has to send the queued [TransactionEventRequest](#) messages. These may contain transactions that were authorized *offline*, as explained in [transaction-related message handling](#). Please refer to [C15 - Unknown Offline Authorization](#) for the options that the Charging Station has to continue / stop the transaction in this situation.

2. Use cases & Requirements

2.1. Authorization options

C01 - EV Driver Authorization using RFID

Table 59. C01 - EV Driver Authorization using RFID

No.	Type	Description
1	Name	EV Driver Authorization using RFID
2	ID	C01
	Functional block	C. Authorization
3	Objective(s)	To enable the Charging Station to request the CSMS to authorize an EV Driver to start or stop charging.
4	Description	When a Charging Station needs to charge an EV, it needs to authorize the EV Driver first before the charging can be started or stopped.
	Actors	Charging Station, CSMS, EV Driver
	Scenario description	<p>1. The EV Driver wants to start or stop charging the EV and presents an RFID card.</p> <p>2. The Charging Station sends AuthorizeRequest to the CSMS to request authorization.</p> <p>3. Upon receipt of AuthorizeRequest, the CSMS responds with AuthorizeResponse. This response message indicates whether or not the IdToken is accepted by the CSMS.</p>
	Alternative scenario(s)	C02 - Authorization using a start button C03 - Authorization using credit/debit card C04 - Authorization using PIN-code C05 - Authorization for CSMS initiated transactions C06 - Authorization using local id type C07 - Authorization using Contract Certificates C08 - Authorization at EVSE using ISO 15118 External Identification Means (EIM) C15 - Unknown Offline Authorization
5	Prerequisite(s)	n/a
6	Postcondition(s)	<p>Successful postcondition: The EV Driver is authorized and can start or stop charging.</p> <p>Failure postcondition: If the authorize message is <i>Invalid</i>, <i>Blocked</i>, <i>Expired</i> or <i>Unknown</i>, the EV Driver can <i>not</i> start or stop charging, except in the case where the EV Driver presents the same token used to start the transaction.</p>

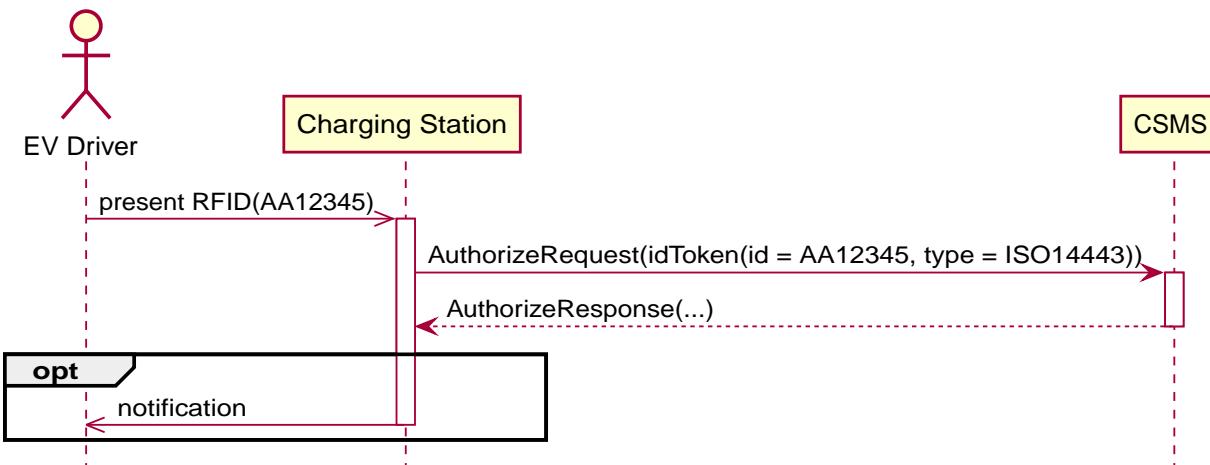


Figure 22. Sequence Diagram: EV Driver Authorization

7	Error handling	When the Authorization is not 'Accepted', the AuthorizeResponse contains an authorization status value indicating the reason for rejection.
---	----------------	---

8	Remark(s)	Assuming <i>idToken</i> is valid for charging and the Charging Station has 3 EVSEs, what is the content of <i>idTokenInfo</i> , when <i>idToken</i> is allowed to charge: .. at all EVES: <i>idTokenInfo.status</i> = Accepted. .. at EVSE 1: <i>idTokenInfo.status</i> = Accepted, <i>idTokenInfo.evsId</i> = [1]. .. at EVSE 1 + 2: <i>idTokenInfo.status</i> = Accepted, <i>idTokenInfo.evsId</i> = [1, 2]. .. at none of the EVSEs: <i>_idTokenInfo.status</i> =NotAtThisLocation.
---	------------------	--

C01 - EV Driver Authorization using RFID - Requirements

Table 60. C01 - Requirements

ID	Precondition	Requirement definition	Note
C01.FR.01	Configuration setting AuthEnabled is true.	The Charging Station SHALL only offer energy after authorization.	
C01.FR.02	If an idToken presented by the EV Driver is not present in the Local Authorization List or Authorization Cache	The Charging Station SHALL send AuthorizeRequest to the CSMS to request authorization.	
C01.FR.03	When an idToken is presented during a transaction that has been authorized AND (a) the presented idToken is the same as the idToken that started the authorization OR (b) when the presented idToken is in the Local Authorization List or Authorization Cache AND is valid AND has the same GroupIdToken as the IdToken that started the authorization.	The Charging Station SHALL end the authorization of the transaction, without first sending an AuthorizeRequest	The idToken that started the authorization can always be used to end the authorization. Ending authorization will end delivery of energy. Depending on the TxStopPoint ending of the authorization may also end the transaction.
C01.FR.04		AuthorizeRequest SHALL only be used for the authorization of an identifier.	
C01.FR.05	If an IdToken is present in the Local Authorization List or Authorization Cache .	The Charging Station MAY send AuthorizeRequest to the CSMS.	
C01.FR.06	When CSMS receives an AuthorizeRequest for an idToken AND the idToken has an associated groupIdToken .	AuthorizeResponse sent by the CSMS to a Charging Station SHALL include the associated groupIdToken .	
C01.FR.07		AuthorizeResponse SHALL include an authorization status value indicating acceptance or a reason for rejection.	See AuthorizationStatusEnumType for the possible reasons of rejection.
C01.FR.08	If the field: language1 is set AND the Charging Station contains messages in that <i>language</i> .	The Charging Station SHALL show messages to the user in language1 .	
C01.FR.09	If the field: language1 is set AND the Charging Station does not contain messages in that <i>language</i> AND if the field: language2 is set AND the Charging Station contains messages in that <i>language</i>	The Charging Station SHALL show messages to the user in language2 .	
C01.FR.10	If the field: language1 is not set	The field: language2 SHALL NOT be set.	
C01.FR.11		Field: language1 SHALL be different from field language2 .	
C01.FR.12		It is RECOMMENDED to implement messages in English as fall-back.	
C01.FR.13	If both language1 AND language2 don't match installed languages in the Charging Station	It is RECOMMENDED to show messages to the EV Driver in English .	
C01.FR.17		Language SHALL be specified as RFC-4646 tags, see: [RFC5646] , example: US English is: "en-US".	
C01.FR.18	If the IdToken is valid AND the EV driver is NOT allowed to charge at the type of EVSE(s) this Charging Station provides.	The CSMS SHALL send an AuthorizeResponse with idTokenInfo.status NotAllowedTypeEVSE .	

ID	Precondition	Requirement definition	Note
C01.FR.19	<i>idToken</i> is allowed for any EVSE of the Charging Station	The CSMS SHALL send an AuthorizeResponse in which <i>IdTokenInfo</i> has an empty (or absent) <i>evsId</i> list.	This will be the most common case. Even though the <i>idToken</i> might be allowed on any EVSE, the <i>idTokenInfo.status</i> still needs to be Accepted before charging is allowed.
C01.FR.20	<i>idToken</i> is allowed for a subset of EVSEs of the Charging Station	The CSMS SHALL send an AuthorizeResponse in which <i>IdTokenInfo</i> has an <i>evsId</i> list with the allowed EVSEs.	Note the difference between validity of an <i>idToken</i> and the fact whether this (type of) token is allowed on an EVSE. The <i>idTokenInfo.status</i> still needs to be Accepted before charging is allowed.
C01.FR.21	C01.FR.20	The Charging Station SHALL only allow charging on the EVSEs mentioned in the AuthorizeResponse.	
C01.FR.22	<i>idToken</i> is not allowed for any EVSE of the Charging Station	The CSMS SHALL send an AuthorizeResponse in which <i>idTokenInfo.status</i> is NotAtThisLocation and <i>evsId</i> list is empty (or absent).	Status NotAtThisLocation needed in order to differentiate with the situation in which <i>idToken</i> is allowed on all EVSEs.
C01.FR.23	When a transaction is still active, that had been authorized earlier by an <i>idToken</i> , but which is now no longer authorized for charging AND a new <i>idToken</i> is presented to the Charging Station for authorization, that differs from the initial <i>idToken</i>	The Charging Station SHOULD not allow the authorization of a different <i>idToken</i> .	Multiple <i>idTokens</i> for a transaction are most likely not supported by a CSMS.
C01.FR.24	When a transaction is still active, that had been authorized earlier by an <i>idToken</i> , but which is now no longer authorized for charging AND Charging Stations sends an AuthorizeRequest for a new <i>idToken</i> , that differs from the initial <i>idToken</i> of the transaction	The CSMS is RECOMMENDED to respond with an AuthorizeResponse with <i>idTokenInfo.status</i> = NotAtThisTime for this <i>idToken</i> .	If a second authorization is done by Charging Station then CSMS can reject the <i>idToken</i> .

C02 - Authorization using a start button

Table 61. C02 - Authorization using a start button

No.	Type	Description
1	Name	Authorization using a start button
2	ID	C02
	Functional block	C. Authorization
3	Objectives	Make it possible for a Charging Station that has a start button to start charging.
4	Description	For some chargers authorization of a user might not be a requirement. A simple charger might have a button instead of a more expensive RFID reader to start charging. When such a Charging Station starts charging, it is not needed to send an AuthorizeRequest . In the TransactionEventRequest (<code>eventType = Started</code>), <code>IdTokenType</code> information needs to be given, which the CSMS then cannot reject.
	Actors	EV Driver, Charging Station, CSMS
	Scenario description	<ol style="list-style-type: none"> 1. The EV Driver plugs in the charging cable between EV and Charging Station. 2. The Charging Station sends a StatusNotificationRequest and TransactionEventRequest (<code>eventType = Started</code>) to notify the CSMS about the cable being plugged in. 3. The EV Driver presses the start button to start Charging. 4. The Charging Station starts Charging of the EV. 5. The Charging Station sends a TransactionEventRequest (<code>eventType = Updated</code>) message with <code>IdTokenEnumType: NoAuthorization</code> to the CSMS to notify the CSMS of the charging that has started. 6. Upon receipt of TransactionEventRequest (<code>eventType = Updated</code>), the CSMS responds with TransactionEventResponse with: <code>IdTokenInfo.status</code> set to <code>Accepted</code>
	Alternative scenario(s)	C01 - EV Driver Authorization using RFID C03 - Authorization using credit/debit card C04 - Authorization using PIN-code C05 - Authorization for CSMS initiated transactions C06 - Authorization using local id type C07 - Authorization using Contract Certificates C08 - Authorization at EVSE using ISO 15118 External Identification Means (EIM) C15 - Unknown Offline Authorization
5	Prerequisites	Charging Station has a start button, instead of an RFID reader to start charging of an EV.
6	Postcondition(s)	Transaction ongoing on Charging Station, CSMS is aware of transaction.

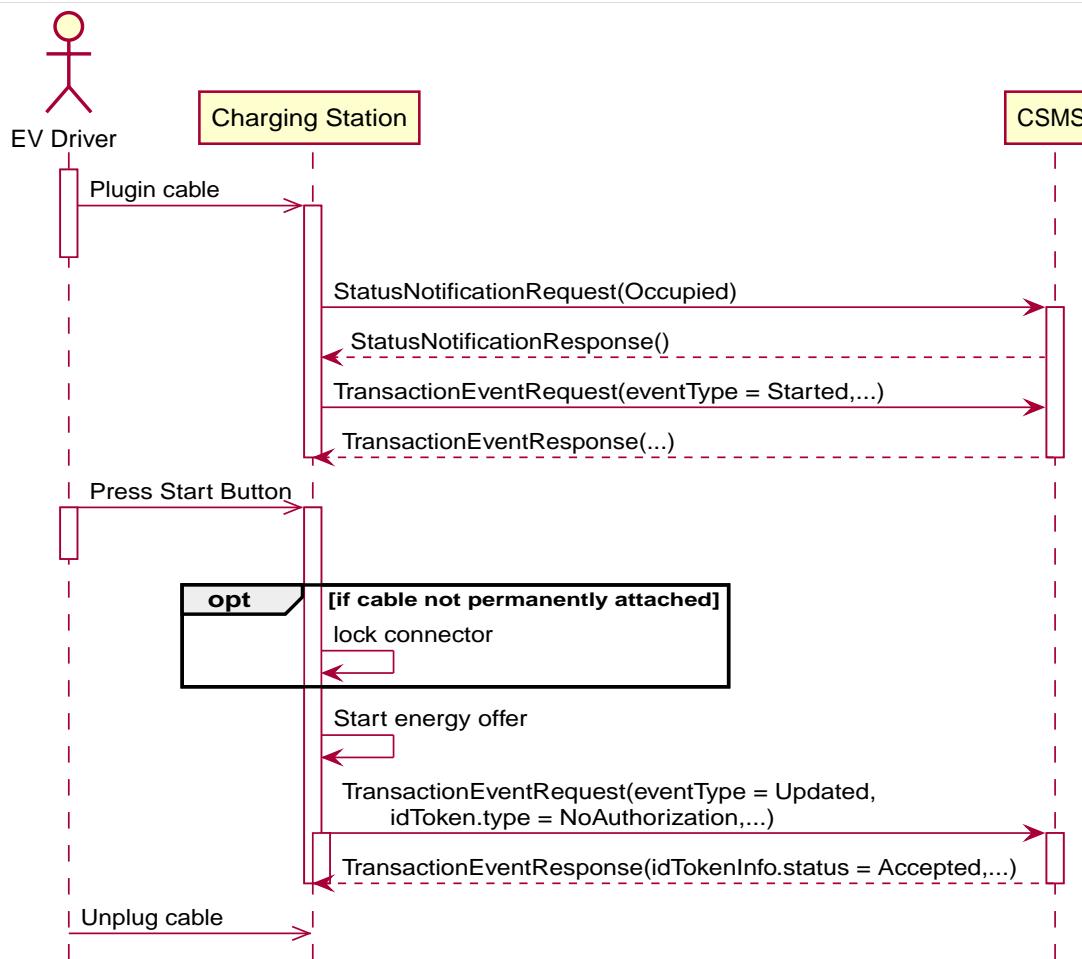


Figure 23. Sequence Diagram: Authorization using a start button

7	Error Handling	n/a
8	Remarks	<p>The start button might also be a mechanical key or something similar.</p> <p>Note that the start button can even be omitted if the Charging Station is configured to start charging upon cable connection.</p> <p>The scenario description and sequence diagram above are based on the Configuration Variable for start transaction being configured as follows:</p> <p>TxStartPoint: EVConnected, Authorized, DataSigned, PowerPathClosed, EnergyTransfer</p> <p>This use-case is also valid for other configurations, but then the transaction might start/stop at another moment, which might change the sequence in which message are send. For more details see the use case: E01 - Start Transaction options.</p>

C02 - Authorization using a start button - Requirements

Table 62. C02 - Authorization using a start button - Requirements

ID.	Precondition	Requirement definition
C02.FR.01	When a transaction is started with a button.	The Charging Station SHALL send TransactionEventRequest with an IdTokenType of type: NoAuthorization and the field: idToken left empty (empty string).
C02.FR.02	CSMS receives a TransactionEventRequest with an IdTokenType of type: NoAuthorization	The CSMS SHALL respond with a TransactionEventResponse with IdTokenInfo.status set Accepted .
C02.FR.03	If the Charging Station has implemented an Authorization Cache AND the Charging Station receives IdTokenInfo for an IdTokenType of type NoAuthorization in any message	The Charging Station SHALL NOT store the information in its Authorization Cache.

C03 - Authorization using credit/debit card

Table 63. C03 - Authorization using credit/debit card

No.	Type	Description
1	Name	Authorization using credit card
2	ID	C03
	<i>Functional block</i>	C. Authorization
3	Objectives	Make it possible to start a transaction using a credit card.
4	Description	A Charging Station with a credit/debit card terminal built inside the housing, or belonging to a group of Charging Stations that has a central payment terminal/kiosk. An EV Driver uses his card to pay for charging. The transaction is authorized by the payment company, the CSMS receives a message from the Payment System, and send a RequestStartTransactionRequest to the Charging Station to start the transaction.
	Actors	EV Driver, Payment System, CSMS, Charging Station
	<i>Scenario description</i>	<ol style="list-style-type: none"> 1. The EV Driver plugs in the Charging Cable 2. The Charging Station sends an StatusNotificationRequest and TransactionEventRequest (<code>eventType = Started</code>) to notify the CSMS about the cable being plugged in. 3. The Driver uses the credit/debit card terminal to authorize/pay for charging. 4. The terminal communicates with its own server/back-office. 5. The Payment System sends a message to the CSMS authorizing the user. 6. The CSMS generates a unique id to be used as IdToken for this transaction. 7. The CSMS sends a RequestStartTransactionRequest with the generated IdToken to the Charging Station. 8. The Charging Station accepts the RequestStartTransactionRequest by sending a RequestStartTransactionResponse with Accepted. 9. The Charging Station start Charging of the EV. 10. The Charging Station send an TransactionEventRequest (<code>eventType = Updated</code>) to notify the CSMS about the charging having started.
	<i>Alternative scenario(s)</i>	C01 - EV Driver Authorization using RFID C02 - Authorization using a start button C04 - Authorization using PIN-code C05 - Authorization for CSMS initiated transactions C06 - Authorization using local id type C07 - Authorization using Contract Certificates C08 - Authorization at EVSE using ISO 15118 External Identification Means (EIM) C15 - Unknown Offline Authorization
5	Prerequisites	Charging Station has a credit/debit card terminal, or belongs to a group of Charging Stations that has a central payment terminal, to start charging of an EV.
6	Postcondition(s)	Transaction ongoing on Charging Station

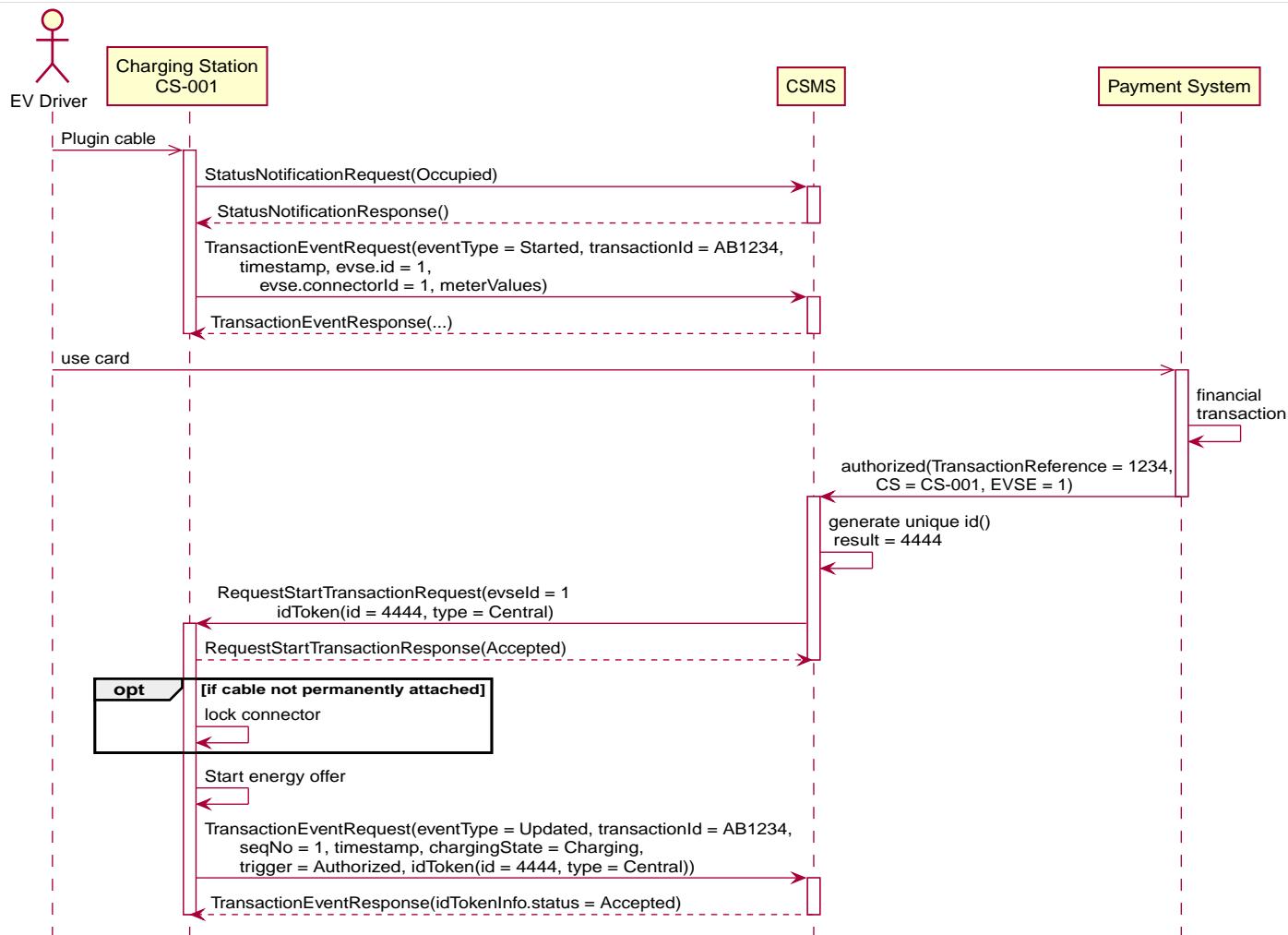


Figure 24. Sequence Diagram: Authorization using credit/debit card

7	Error Handling	n/a
8	Remarks	<p>This use case is an example of how the existing OCPP messages can be used to handle a transaction that is started with a credit/debit card, it is not required to implement a credit/debit card payment solution in this way.</p> <p>A Payment System may consist of multiple components handling the authorization of the user. The interface of these components and the communication between the Payment System and CSMS are not in scope of this document.</p> <p>Stopping a transaction started with a credit/debit card is not defined, this is left to the implementer, this could for example be: Unplugging the cable on the EV side and/or a stop button etc.</p> <p>The scenario description and sequence diagram above are based on the Configuration Variable for start transaction being configured as follows:</p> <p>TxStartPoint: <code>EVConnected</code>, <code>Authorized</code>, <code>DataSigned</code>, <code>PowerPathClosed</code>, <code>EnergyTransfer</code></p> <p>This use-case is also valid for other configurations, but then the transaction might start/stop at another moment, which might change the sequence in which message are send. For more details see the use case: E01 - Start Transaction options.</p>

C03 - Authorization using credit/debit card - Requirements

Table 64. C03 - Authorization using credit/debit card - Requirements

ID.	Precondition	Requirement definition
C03.FR.01	If the Charging Station receives a <code>RequestStartTransactionRequest</code> with an <code>IdTokenType</code> of type <code>Central</code>	The Charging Station SHALL NOT send an <code>AuthorizeRequest</code> for the received <code>IdTokenType</code> .

ID.	Precondition	Requirement definition
C03.FR.02	If the Charging Station has implemented an Authorization Cache AND the Charging Station receives IdTokenInfo for an IdTokenType of type Central in any message	The Charging Station SHALL NOT store the information in its Authorization Cache.

C04 - Authorization using PIN-code

This is an informative use case, its purpose is to demonstrate the use of the [KeyCode](#) id type. An other use of [KeyCode](#) is for example a licence plate number.

Table 65. C04 - Authorization using PIN-code

No.	Type	Description
1	Name	Authorization using PIN-code
2	ID	C04
	<i>Functional block</i>	C. Authorization
3	Objectives	To make it possible for a Charging Station that has a key entry terminal to authorize the PIN-code.
4	Description	When a Charging Station has a PIN-code entry terminal, an EV driver enters his/her PIN-code. This PIN-code is sent to the CSMS for validation using an AuthorizeRequest .
	Actors	EV Driver, Charging Station, CSMS
	<i>Scenario description</i>	<ol style="list-style-type: none"> 1. The EV Driver wants to start or stop charging the EV and enters his/her PIN-code into the terminal. 2. The Charging Station sends an AuthorizeRequest message, with the field: <code>IdTokenEnumType</code> set to KeyCode, to the CSMS to request authorization. 3. Upon receipt of the AuthorizeRequest, the CSMS responds with an AuthorizeResponse. This response indicates whether or not the KeyCode is accepted by the CSMS.
	<i>Alternative scenario(s)</i>	C01 - EV Driver Authorization using RFID C02 - Authorization using a start button C03 - Authorization using credit/debit card C05 - Authorization for CSMS initiated transactions C06 - Authorization using local id type C07 - Authorization using Contract Certificates C08 - Authorization at EVSE using ISO 15118 External Identification Means (EIM) C15 - Unknown Offline Authorization
5	Prerequisites	Charging Station has a PIN-code entry terminal to start charging of an EV.
6	Postcondition(s)	Transaction ongoing on Charging Station, CSMS is aware of transaction.

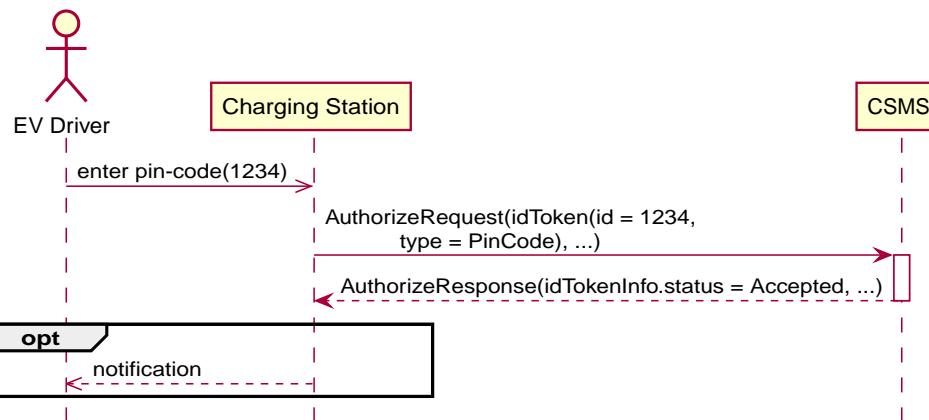


Figure 25. Sequence Diagram: Authorization using PIN-code

7	Error Handling	n/a
8	Remarks	When the PIN-code is validated in the Charging Station, instead of the CSMS, use case C02 - Authorization Using a Start button applies.

C04 - Authorization using PIN-code - Requirements

Table 66. C04 - Authorization using PIN-code - Requirements

ID.	Precondition	Requirement definition
C04.FR.01	When the CSMS receives an AuthorizeRequest with a KeyCode that is not valid at this Charging Station	The CSMS SHALL respond with an AuthorizeResponse message with <code>status = Invalid</code> .

ID.	Precondition	Requirement definition
C04.FR.02	When the CSMS receives an AuthorizeRequest with a keyCode that is valid and the EV Driver is allowed to charge at this Charging Station	The CSMS SHALL respond with an AuthorizeResponse message with status = Accepted .
C04.FR.03		A Charging Station MAY store keyCodes in the Authorization Cache.
C04.FR.04	If an idToken of type keyCode is used	The Charging Station or CSMS SHALL NOT show the IdToken in any logging. key codes should never appear in logs.
C04.FR.05		Language SHALL be specified as RFC-5646 tags, see: [RFC5646] , for example: US English is: "en-US".
C04.FR.06	If an idToken of type keyCode is used	It is RECOMMENDED to take measures to prevent brute force attacks, for example by increasing backoff times after attempts to enter an incorrect keyCode.

C05 - Authorization for CSMS initiated transactions

Table 67. C05 - Authorization for CSMS initiated transactions

No.	Type	Description
1	Name	Authorization for CSMS initiated transactions
2	ID	C05
	Functional block	C. Authorization
3	Objectives	Enable the CSMS to start a transaction on a Charging Station with a server generated IdToken.
4	Description	When a CSMS needs to start a Transaction on a Charging Station for a Driver that has no RFID, or the RFID is not known. For Example, the EV Driver uses an App to start a transaction. The CSMS needs to determine an IdToken and tell the Charging Station this is not an RFID, so it should not be cached and an authorization is also not needed.
	Actors	EV Driver, CSMS, Charging Station
	Scenario description	<ol style="list-style-type: none"> 1. The EV Driver uses his app to start a charging. 2. The app sends a start request to the CSMS. 3. The CSMS determines an IdToken. It can generate a unique id to be used as IdToken for this transaction or can use a token that is provided by the app (for example the ID of the contract of the user). 4. The CSMS sends a RequestStartTransactionRequest with the IdToken from the previous step to the Charging Station. 5. The Charging Station accepts the RequestStartTransactionRequest by sending a RequestStartTransactionResponse with Accepted. 6. The Charging Station starts charging and sends a TransactionEventRequest (eventType = Updated) to notify the CSMS that chargingState has changed.
	Alternative scenario(s)	C01 - EV Driver Authorization using RFID C02 - Authorization using a start button C03 - Authorization using credit/debit card C04 - Authorization using PIN-code C06 - Authorization using local id type C07 - Authorization using Contract Certificates C08 - Authorization at EVSE using ISO 15118 External Identification Means (EIM) C15 - Unknown Offline Authorization
5	Prerequisites	Cable is plugged in.
6	Postcondition(s)	Transaction ongoing on Charging Station

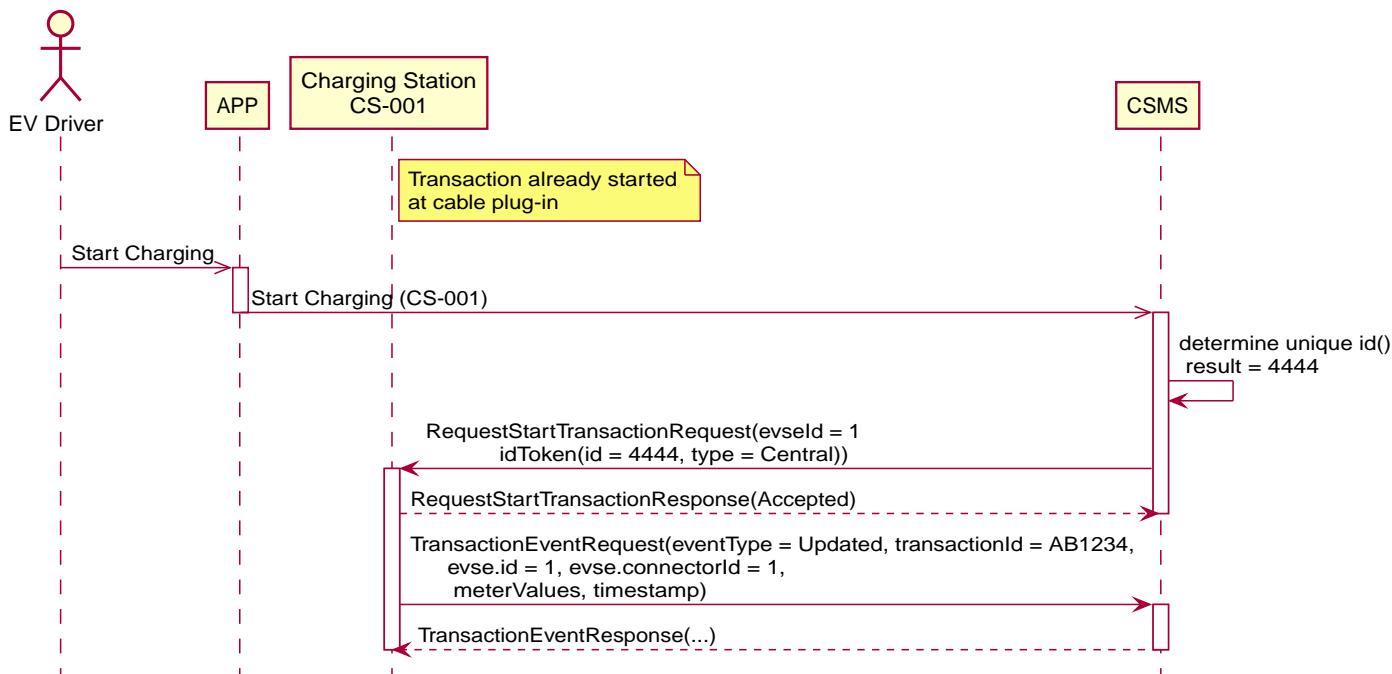


Figure 26. Sequence Diagram: Authorization for CSMS initiated transactions

7	Error Handling	n/a
8	Remarks	<p>IdTokens MAY be (single use) virtual transaction authorization codes or virtual RFID tokens that deliberately use a non-standard UID format to avoid possible conflict with real UID values. These virtual single use IdTokens are sent with type Central and it is pointless to either cache or authorize these tokens.</p> <p>This use case uses an App as example, but this is not a requirement. This use case is valid for any RequestStartTransactionRequest with a server generated IdToken.</p> <p>The scenario description and sequence diagram above are based on the Configuration Variable for start transaction being configured as follows:</p> <p>TxStartPoint: EVConnected, Authorized, DataSigned, PowerPathClosed, EnergyTransfer This use-case is also valid for other configurations, but then the transaction might start/stop at another moment, which might change the sequence in which message are send. For more details see the use case: E01 - Start Transaction options.</p> <p>This use case assumes that the configuration variable AuthorizeRemoteStart is false. See use cases F01 and F02 for requirements with AuthorizeRemoteStart.</p> <p>Other idTokenType can also be used to remote start charging, such an eMAID of the user that is provided by the app.</p>

C05 - Authorization for CSMS initiated transactions Requirements

Table 68. C05 - Authorization for CSMS initiated transactions Requirements

ID.	Precondition	Requirement definition
C05.FR.01	If the Charging Station receives a RequestStartTransactionRequest with an IdTokenType of type Central.	The Charging Station SHALL NOT send an AuthorizeRequest for the received IdTokenType.
C05.FR.02	If the Charging Station has implemented an Authorization Cache AND the Charging Station receives IdTokenInfo for an IdTokenType of type Central in any message	The Charging Station SHALL NOT store the information in its Authorization Cache.
C05.FR.03		The RemoteStartId SHALL be provided at least once in a TransactionEventRequest.
C05.FR.04		Language SHALL be specified as RFC-4646 tags, see: [RFC5646], example: US English is: "en-US".
C05.FR.05		idToken SHALL also be provided once in the first TransactionEventRequest after a RequestStartTransactionRequest.

C06 - Authorization using local id type

This is an informative use case, its purpose is to demonstrate the use of the [Local](#) id type.

Table 69. C06 - Authorization using local id type

No.	Type	Description
1	Name	Authorization using local id type
2	ID	C06
	<i>Functional block</i>	C. Authorization
3	Objectives	Enable the Charging Station to start charging with a locally generated IdToken.
4	Description	When a Charging Station needs to start a Transaction for a Driver that has no RFID, or the RFID is not known. For Example, the EV Driver uses a parking ticket to start charging.
	Actors	EV Driver, Payment Terminal, CSMS, Charging Station
	<i>Scenario description</i>	<ol style="list-style-type: none"> 1. An EV driver drives into a garage, takes a parking ticket at the barrier at the entrance. 2. Parks his EV at a Charging Station. 3. Plugs in the charging cable. 4. Scans/inserts his parking ticket on the Charging Station to start Charging 5. EV is charging, driver leaves. 6. EV driver returns, inserts parking ticket into a payment kiosk 7. Pays for parking and charging 8. The Payment terminal/kiosk sends a stop command via the CSMS to the Charging Station. 9. EV driver unplugs the charging cable and drives away.
	<i>Alternative scenario(s)</i>	C01 - EV Driver Authorization using RFID C02 - Authorization using a start button C03 - Authorization using credit/debit card C04 - Authorization using PIN-code C05 - Authorization for CSMS initiated transactions C07 - Authorization using Contract Certificates C08 - Authorization at EVSE using ISO 15118 External Identification Means (EIM) C15 - Unknown Offline Authorization
5	Prerequisites	Integrated parking & charging payment system
6	Postcondition(s)	The transaction has completed at the Charging Station and Transaction information is available at the CSMS.

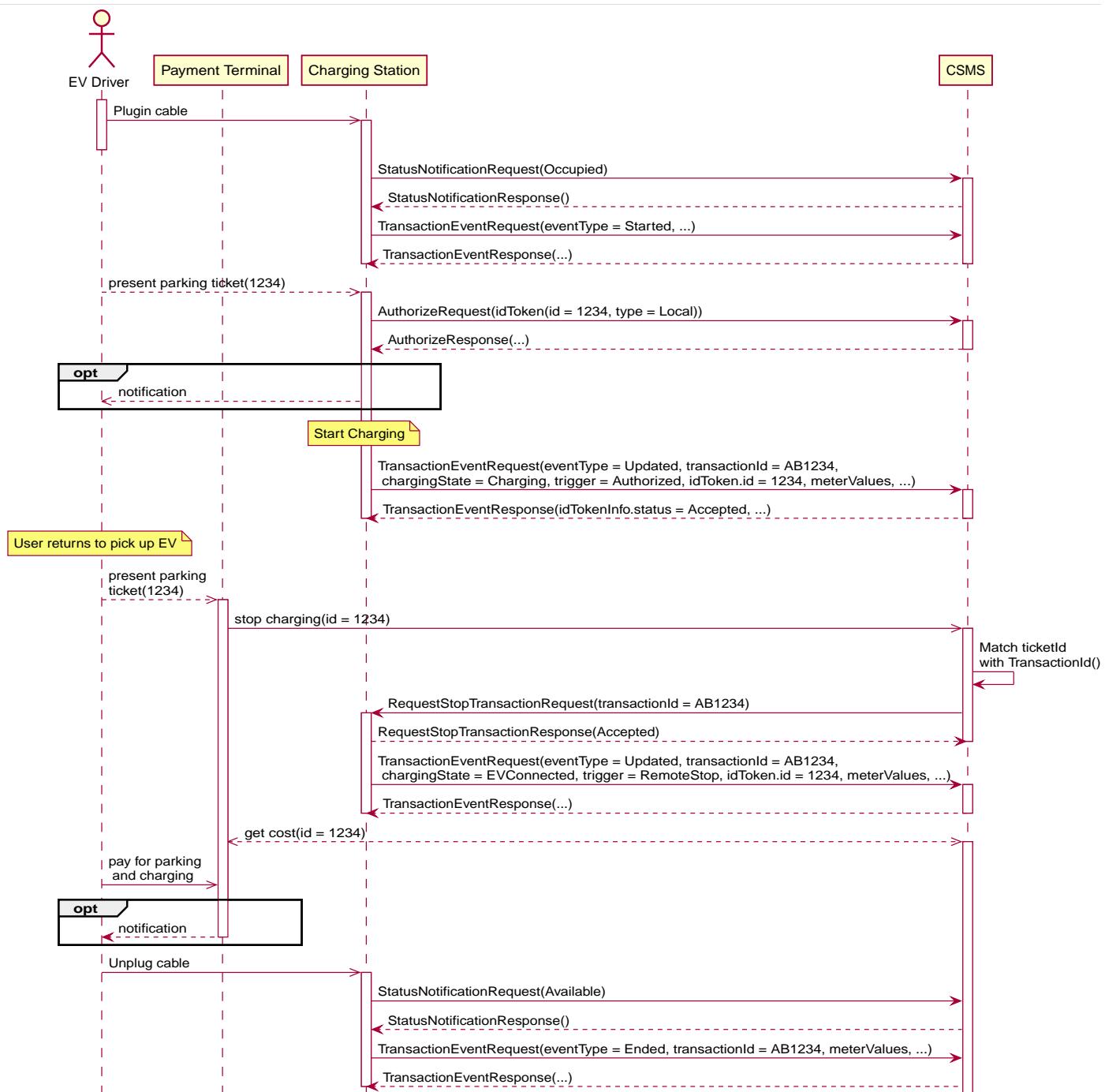


Figure 27. Sequence Diagram: Authorization using local id type

7	Error Handling	n/a
8	Remarks	<p>This use case uses an Parking Ticket as example, but this is not a requirement.</p> <p>The communication between the Payment Terminal and the CSMS is outside of scope of OCPP.</p> <p>The scenario description and sequence diagram above are based on the Configuration Variable for start & stop transaction being configured as follows:</p> <p>TxStartPoint: Authorized, DataSigned, PowerPathClosed, EnergyTransfer</p> <p>TxStopPoint: ParkingBayOccupancy, EVConnected</p> <p>This use-case is also valid for other configurations, but then the transaction might start/stop at another moment, which might change the sequence in which message are send. For more details see the use cases: E01 - Start Transaction options and E06 - Stop Transaction options.</p>

C06 - Authorization using local id type - Requirements

Table 70. C06 - Authorization using local id type - Requirements

ID	Precondition	Requirement definition
C06.FR.01		The Charging Station SHALL only offer energy after authorization.
C06.FR.02	If an IdTokenType with type Local is presented by the EV Driver.	The Charging Station SHALL send AuthorizeRequest to the CSMS to request authorization.
C06.FR.03		AuthorizeRequest SHOULD only be used for the authorization of an identifier for charging.
C06.FR.04	If the CSMS receives an AuthorizeRequest .	it SHALL respond with an AuthorizeResponse and SHALL include an authorization status value indicating acceptance or a reason for rejection.

2.2. ISO 15118 Authorization

This authorization section originates from [ISO15118-1](#) for the use of Plug & Charge functionalities.

C07 - Authorization using Contract Certificates

Table 71. C07 - Authorization using Contract Certificates

No.	Type	Description
1	Name	Authorization using Contract Certificates
2	ID	C07
	<i>Functional block</i>	C. Authorization
	<i>Reference</i>	ISO15118-1 D2
3	Objectives	See ISO15118-1 , use case Objective D2, page 26.
4	Description	See ISO15118-1 , use case Description D2 (first bullet), page 26.
	<i>Actors</i>	Actors: EV, Charging Station, CSMS, OCSP
	<i>Scenario description</i>	<p>15118: See ISO15118-1, use case Description D2, Scenario Description, first 2 bullets, page 26.</p> <p>OCPP:</p> <ul style="list-style-type: none"> 3. The Charging Station sends an AuthorizeRequest message to the CSMS containing the eMAID and data needed for an OCSP request with regards to the contract certificate and certificate chain. 4. The CSMS replies with an agreement or non-agreement, and the certificate status. 5. Service starts after successful authorization of the IDs.
	<i>Alternative scenario(s)</i>	C01 - EV Driver Authorization using RFID C02 - Authorization using a start button C03 - Authorization using credit/debit card C04 - Authorization using PIN-code C05 - Authorization for CSMS initiated transactions C06 - Authorization using local id type C08 - Authorization at EVSE using ISO 15118 External Identification Means (EIM) C15 - Unknown Offline Authorization
5	Prerequisites	A contract Certificate is installed in the EV.
6	Postcondition(s)	The validity of the Contract Certificate is determined.

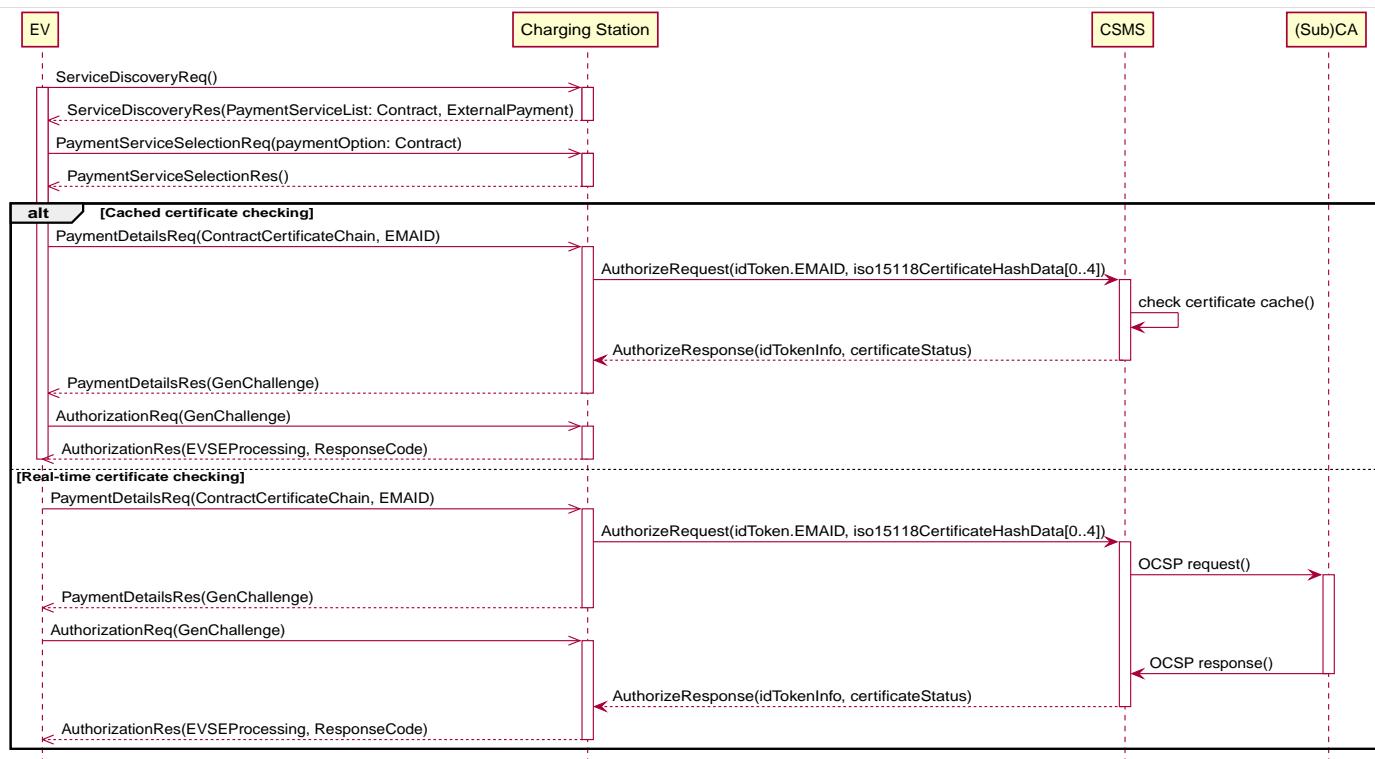


Figure 28. Authorization using Contract Certificates

7	Error handling	
8	Remark(s)	<p>In edition 1 of 15118, the message timeout of the <code>PaymentDetailsReq/Res</code> message is 5 seconds. In case certificate verification cannot be completed in that time it is possible to complete this during the <code>AuthorizationReq/Res</code>, which can be extended up to 60 seconds.</p> <p>When the Charging Station is offline, it is recommended to omit the payment option for ISO 15118 contract certificates from the <code>ServiceDiscoveryRes</code> and revert to External Identification Means (use case C08), because certificate status cannot be checked.</p>

C07 - Authorization using Contract Certificates - Requirements

Table 72. C07 - Requirements

ID	Precondition	Requirement definition
C07.FR.01	When Charging Station is online	The Charging Station SHALL send an <code>AuthorizeRequest</code> to the CSMS for validation.
C07.FR.02	C07.FR.01	The <code>AuthorizeRequest</code> SHALL contain the eMAID and data needed for an OCSP request with regards to the contract certificate and certificate chain.
C07.FR.04	If the CSMS receives an <code>AuthorizeRequest</code> .	It SHALL respond with an <code>AuthorizeResponse</code> and SHALL include an authorization status value indicating acceptance or a reason for rejection.
C07.FR.05	C07.FR.02	The CSMS SHALL verify validity of the certificate and certificate chain via real-time or cached OCSP data using the hash data provided in <code>iso15118CertificateHashData</code> field.
C07.FR.06	C07.FR.01 AND If Charging Station is not able to validate a contract certificate, because it does not have the associated root certificate AND <code>CentralContractValidationAllowed</code> is true	The Charging Station SHALL pass the contract certificate chain to the CSMS in <code>certificate</code> attribute (in PEM format) of <code>AuthorizeRequest</code> for validation by CSMS.
C07.FR.07	When Charging Station is offline AND <code>ContractValidationOffline</code> is false	The Charging Station SHALL NOT allow charging.
C07.FR.08	When Charging Station is offline AND <code>ContractValidationOffline</code> is true	The Charging Station SHALL try to validate the contract certificate locally.

ID	Precondition	Requirement definition
C07.FR.09	C07.FR.08 AND Contract certificate is valid AND <code>LocalAuthorizeOffline</code> is <code>true</code>	The Charging Station SHALL lookup the eMAID in Local Authorization List or Authorization Cache .
C07.FR.10	C07.FR.09 AND eMAID found in Local Authorization List	The Charging Station SHALL behave according to use case C13 - Offline Authorization through Local Authorization List .
C07.FR.11	C07.FR.09 AND eMAID found in Authorization Cache	The Charging Station SHALL behave according to use case C12 - Start Transaction - Cached Id .
C07.FR.12	C07.FR.09 AND eMAID is not found AND <code>OfflineTxForUnknownIdEnabled</code> = <code>true</code>	The Charging Station SHALL allow charging according to use case C15 - Offline Authorization of unknown Id .

C08 - Authorization at EVSE using ISO 15118 External Identification Means (EIM)

Table 73. C08 - Authorization at EVSE using ISO 15118 External Identification Means (EIM)

No.	Type	Description
1	Name	Authorization at EVSE using ISO 15118 External Identification Means (EIM)
2	ID	C08 / 15118-1 D4
	Functional block	C. Authorization
	Reference	ISO15118-1 D4
3	Objectives	To authorize the EV via the Charging Station, with help of the CSMS. Also see ISO15118-1 , use case Objective D4, page 28.
4	Description	The Charging Station sends an AuthorizeRequest message based on information provided by the EV. Also see ISO15118-1 , use case Description D4 up to and including "NOTE", page 28.
	Actors	Actors: EV, Charging Station, CSMS
	Scenario description	<p>15118 See ISO15118-1, use case Description (Scenarion Description) D4, page 28.</p> <p>OCPP</p> <ol style="list-style-type: none"> 1. The Charging Station sends an AuthorizeRequest with an idToken containing the External Identification Means (EIM). 2. The CSMS responds with an AuthorizeResponse.
	Alternative scenario(s)	C01 - EV Driver Authorization using RFID C02 - Authorization using a start button C03 - Authorization using credit/debit card C04 - Authorization using PIN-code C05 - Authorization for CSMS initiated transactions C06 - Authorization using local id type C07 - Authorization using Contract Certificates C15 - Unknown Offline Authorization
5	Prerequisites	Communication between EV and EVSE SHALL be established successfully.
6	Postcondition(s)	Authorization is successful. Also see ISO15118-1 , use case End conditions D4, page 28.

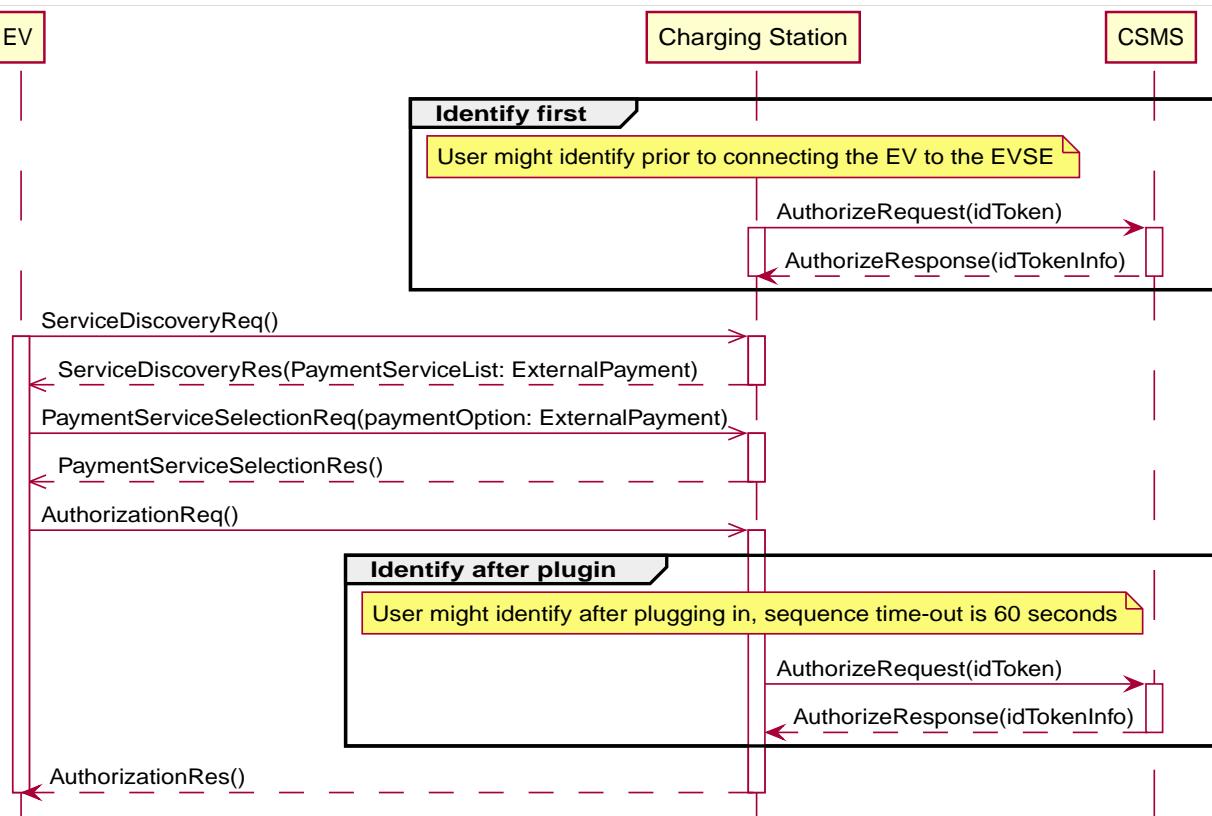


Figure 29. Sequence Diagram: Authorization at EVSE using external credentials performed with help of SA.

7	Remark(s)	Please note that all identification means mentioned in the previous section can be applied to this use case. The only difference is the availability of 15118 communication.
---	-----------	--

Source: [ISO15118-1](#)

C08 - Authorization at EVSE using ISO 15118 External Identification Means (EIM) - Requirements

Table 74. C08 - Requirements

ID	Precondition	Requirement definition
C08.FR.01		The Charging Station SHALL send the identification to the CSMS for validation.
C08.FR.02		EV Driver SHALL activate the authorization within a specific time after connecting the EV to the EVSE or the EVSE SHALL have an HMI to authorize the restart of the identification process.

2.3. GroupId

C09 - Authorization by GroupId

Table 75. C09 - Authorization by GroupId

No.	Type	Description
1	Name	Authorization by GroupId
2	ID	C09
	Functional block	C. Authorization
3	Objective(s)	To enable 2 EV drivers with different IdTokens to be authorized using the same GroupId.
4	Description	This use cases covers how a Charging Station can authorize an action for an EV Driver based on GroupId information. This could for example be used if 2 people regularly use the same EV: they can use their own IdToken (e.g. RFID card), and can deauthorize transactions that were started with the other idToken (with the same GroupId).
	Actors	Charging Station, CSMS, EV Driver1, EV Driver2
	Scenario description	<p>1. EV Driver 1 presents an IdToken.</p> <p>2. The Charging Station sends AuthorizeRequest to the CSMS to request authorization.</p> <p>3. Upon receipt of AuthorizeRequest, the CSMS responds with AuthorizeResponse. This response message includes the GroupId.</p> <p>4. The Charging Station stores the GroupIdToken with the authorization information of EV Driver 1.</p> <p>5. EV Driver 2 presents an IdToken.</p> <p>6. The Charging Station sends AuthorizeRequest to the CSMS to request authorization.</p> <p>7. Upon receipt of AuthorizeRequest, the CSMS responds with AuthorizeResponse. This response message includes the GroupId.</p> <p>8. Based on the matching GroupId information in both responses, the Charging Station authorizes the action.</p>
5	Prerequisite(s)	EV Driver 1 and EV Driver 2 have the same GroupId.
6	Postcondition(s)	GroupId is known by the Charging Station.

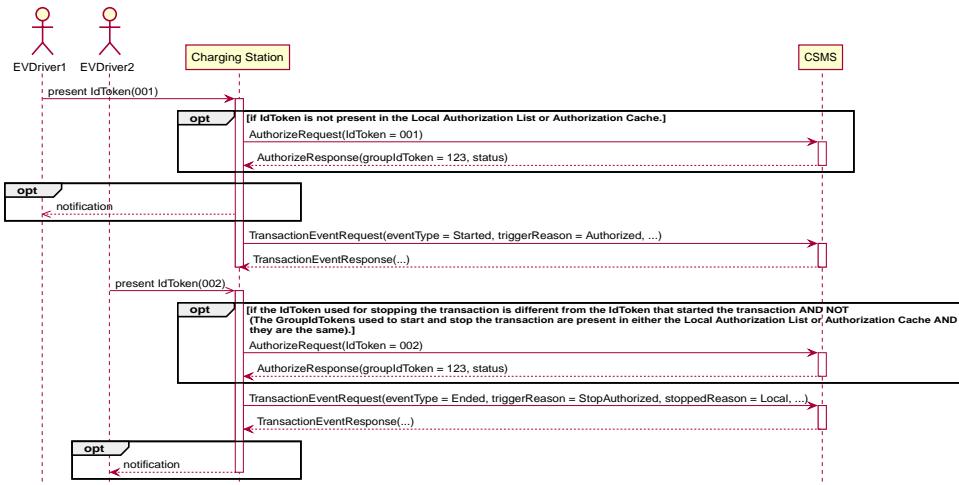


Figure 30. Sequence Diagram: Authorization by GroupId

7	Error handling	n/a
8	Remark(s)	<p>IdTokenType data used as groupId may often use a shared central account identifier for the GroupId, instead of using one of the idTokens belonging to an account.</p> <p>The groupId mechanism as described in this use case also works when using the Authorization Cache, as the groupId is stored in the cache.</p>

C09 - Authorization by GroupId - Requirements

Table 76. C09 - Requirements

ID	Precondition	Requirement definition
C09.FR.02		IdTokens that are part of the same group for authorization purposes SHALL have a common group identifier in the optional <code>groupIdToken</code> element in <code>IdTokenInfo</code>
C09.FR.03	When a transaction has been authorized/started with a certain IdToken.	An EV Driver with a different, valid IdToken, but with the same <code>groupIdToken</code> SHALL be authorized to stop the transaction.
C09.FR.04	C09.FR.03 AND If both IdTokens with their corresponding GroupIdTokens are present in either the Local Authorization List or Authorization Cache .	The Charging Station MAY send an AuthorizeRequest to the CSMS.
C09.FR.05	C09.FR.03 AND (NOT C09.FR.07) AND If the newly presented IdToken with its corresponding GroupIdToken is not present in either the Local Authorization List or Authorization Cache .	The Charging Station SHALL send an AuthorizeRequest to the CSMS.
C09.FR.07	When an <code>idToken</code> is presented during a transaction that has been authorized AND (a) the presented <code>idToken</code> is the same as the <code>idToken</code> that started the authorization OR (b) when the presented <code>idToken</code> is in the Local Authorization List or Authorization Cache AND is valid AND has the same GroupIdToken as the IdToken that started the authorization.	The Charging Station SHALL end the authorization of the transaction, without first sending an AuthorizeRequest
C09.FR.09	If the IdToken in AuthorizeRequest has an associated <code>groupIdToken</code>	AuthorizeResponse from CSMS SHALL include <code>groupIdToken</code> .
C09.FR.10		AuthorizeResponse SHALL include an authorization status value indicating acceptance or a reason for rejection.
C09.FR.11	C09.FR.03 AND A different IdToken is presented for stopping, which has the same GroupIdToken, but does not have status = Accepted	The Charging Station SHALL NOT stop the transaction.
C09.FR.12	If a TransactionEventRequest contains an IdToken and idToken has an associated <code>groupIdToken</code>	TransactionEventResponse from CSMS SHALL include <code>groupIdToken</code> .

2.4. Authorization Cache

C10 - Store Authorization Data in the Authorization Cache

Table 77. C10 - Store Authorization Data in Authorization Cache

No.	Type	Description
1	Name	Store Authorization Data in the Authorization Cache
2	ID	C10
	Functional block	C. Authorization
3	Objective(s)	To store all the latest received IdTokens in the Authorization Cache.
4	Description	This use case covers how the Charging Station autonomously stores a record of previously presented identifiers that have been successfully authorized by the CSMS in the Authorization Cache. (Successfully meaning: a response received on a message containing an IdToken)
	Actors	Charging Station, CSMS
	Scenario description	<p>1. The Charging Station receives a AuthorizeResponse, ReserveNowRequest or TransactionEventResponse response message from the CSMS.</p> <p>2. The Cache is updated by the Charging Station using all received IdTokenInfo from the response message from the CSMS.</p>
	Alternative scenario(s)	n/a
5	Prerequisite(s)	An Authorization Cache is implemented and the value of the AuthCacheEnabled Configuration Variable is set to 'true'.
6	Postcondition(s)	<p>Successful postcondition: The Charging Station stored the newly received IdTokenInfo data in the Authorization Cache.</p> <p>Failure postcondition: The Charging Station was not able to store the Authorization Cache.</p>

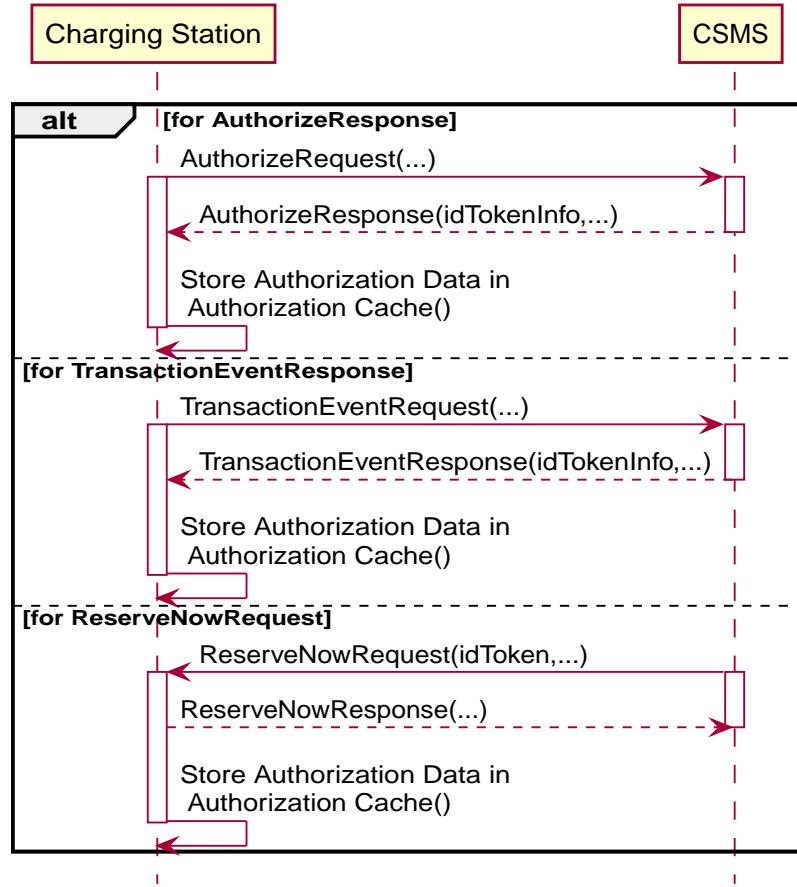


Figure 31. Sequence Diagram: Store Authorization Data in the Authorization Cache

7	Error handling	n/a
---	----------------	-----

8	Remark(s)	n/a
---	-----------	-----

C10 - Store Authorization Data in the Authorization Cache - Requirements

Table 78. C10 - Requirements

ID	Precondition	Requirement definition	Note
C10.FR.01		The Authorization Cache SHALL contain all the latest received identifiers (regardless of their status).	
C10.FR.02		Cache values SHOULD be persistent across reboots and power outages.	Hence cache values SHOULD be stored in non-volatile memory.
C10.FR.03	When an IdToken is presented that is stored in the Authorization Cache with status other than Accepted, and the Charging Station is online.	AuthorizeRequest SHALL be sent to the CSMS to check the current state of the IdToken.	To check the current state of the identifier.
C10.FR.04	Upon receipt of AuthorizeResponse .	The Charging Station SHALL update the Authorisation Cache entry.	The update is to be done with the IdTokenInfo value from the response as described under Authorization Cache .
C10.FR.05	Upon receipt of TransactionEventResponse .	The Charging Station SHALL update the Authorisation Cache entry.	The update is to be done with the IdTokenInfo value from the response as described under Authorization Cache .
C10.FR.06	Upon receipt of ReserveNowRequest .	The Charging Station SHALL update the Authorisation Cache entry.	The update is to be done with the IdTokenInfo value from the request as described under Authorization Cache .
C10.FR.07		The Charging Station SHALL have a mechanism to accept new cache entries even when it is full, by deleting older entries.	It is suggested to remove any entries with status other than Accepted first, and then the oldest valid entries to make space for the new entry.
C10.FR.08		The time a token may live in the cache is determined by the Configuration Variable AuthCacheLifeTime . This variable indicates how long it takes until a token expires in the Authorization Cache since it is last used.	This expiry of the cache is not the same as the expiration date that is set for the IdToken (e.g. RFID card expiry date).
C10.FR.09	The Charging Station supports Tariff & Cost	The Charging Station SHALL NOT store the tariff information in the Cache.	
C10.FR.10	When the validity of an Authorization Cache entry expires.	The Authorization Cache entry SHALL be removed from the cache or changed to Expired .	
C10.FR.11		Whether the Authorization Cache is enabled or disabled SHALL be controlled by the AuthCacheEnabled Configuration Variable.	
C10.FR.12		It is RECOMMENDED to store personal information in the Authorization Cache securely	E.g. by only storing hashed idTokens in the cache.

C11 - Clear Authorization Data in Authorization Cache

Table 79. C11 - Clear Authorization Data in Authorization Cache

No.	Type	Description
1	Name	Clear Authorization Data in Authorization Cache
2	ID	C11
	Functional block	C. Authorization
3	Objective(s)	To clear all IdTokens in the Authorization Cache.
4	Description	This use case covers how the CSMS can request a Charging Station to clear its Authorization Cache.
	Actors	Charging Station, CSMS
	Scenario description	<p>1. The CSMS requests the Charging Station to clear its Authorization Cache by sending ClearCacheRequest.</p> <p>2. The Charging Station responds with the status Accepted.</p>
5	Prerequisite(s)	Authorization Cache is supported and enabled by the AuthCacheEnabled Configuration Variable.
6	Postcondition(s)	<p>Successful postcondition: The Charging Station <i>Successfully cleared</i> the Authorization Cache.</p> <p>Failure postcondition: The Charging Station was <i>not able</i> to clear the Authorization Cache.</p>

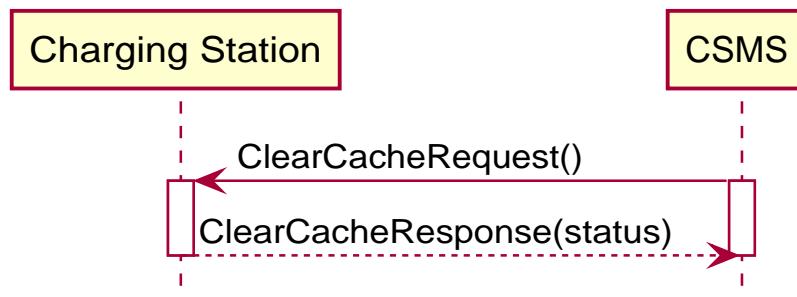


Figure 32. Sequence Diagram: Clear Authorization Data in Authorization Cache

7	Error handling	n/a
8	Remark(s)	n/a

C11 - Clear Authorization Data in Authorization Cache - Requirements

Table 80. C11 - Requirements

ID	Precondition	Requirement definition
C11.FR.01	If the CSMS sends a ClearCacheRequest .	The Charging Station SHALL attempt to clear its Authorization Cache.
C11.FR.02	C11.FR.01	The Charging Station SHALL send ClearCacheResponse message indicating whether it was able to clear its Authorization Cache.
C11.FR.03	C11.FR.02 AND Charging Station successfully cleared its Authorization Cache.	The Charging Station SHALL send ClearCacheResponse message with the status Accepted .
C11.FR.04	C11.FR.02 AND Configuration variable AuthCacheEnabled is false	The Charging Station SHALL send ClearCacheResponse message with the status Rejected .
C11.FR.05	C11.FR.02 AND Charging Station failed to clear its Authorization Cache.	The Charging Station SHALL send ClearCacheResponse message with the status Rejected .

C12 - Start Transaction - Cached Id

Table 81. C12 - Start Transaction - Cached Id

No.	Type	Description
1	Name	Start Transaction - Cached Id
2	ID	C12
	Functional block	C. Authorization
3	Objective(s)	To enable the EV Driver to <i>Online</i> start a transaction by using the Authorization Cache. So the Charging Station can respond faster, as no AuthorizeRequest is being sent.
4	Description	This use case describes how the EV Driver is authorized to start a transaction while the Charging Station uses Cached IdToken.
	Actors	Charging Station, CSMS, EV Driver
	Scenario description	<ol style="list-style-type: none"> 1. The EV Driver plugs in the cable. 2. The Charging Station starts the transaction. 3. The EV Driver presents an IdToken. 4. The Charging Station verifies the IdToken with the Authorization Cache. 5. The Charging Station updates the transaction. 6. The Charging Station starts charging. 7. E02 - Start Transaction - Cable Plugin First applies.
5	Prerequisite(s)	<p><code>AuthCacheEnabled = true</code> <code>LocalPreAuthorize = true</code></p> <p>The Id of the EV Driver is Cached in the Authorization Cache Id is valid</p>
6	Postcondition(s)	<p>Successful postcondition: The EV Driver is authorized to start a transaction by using the Authorization Cache.</p> <p>Failure postcondition: The UserId was not found in the Authorization Cache and: * Online Charging Station: the Charging Station issues an AuthorizeRequest and that fails too. * In an offline situation, behaviour of the Charging Station is defined by Configuration Variable OfflineTxForUnknownIdEnabled.</p>

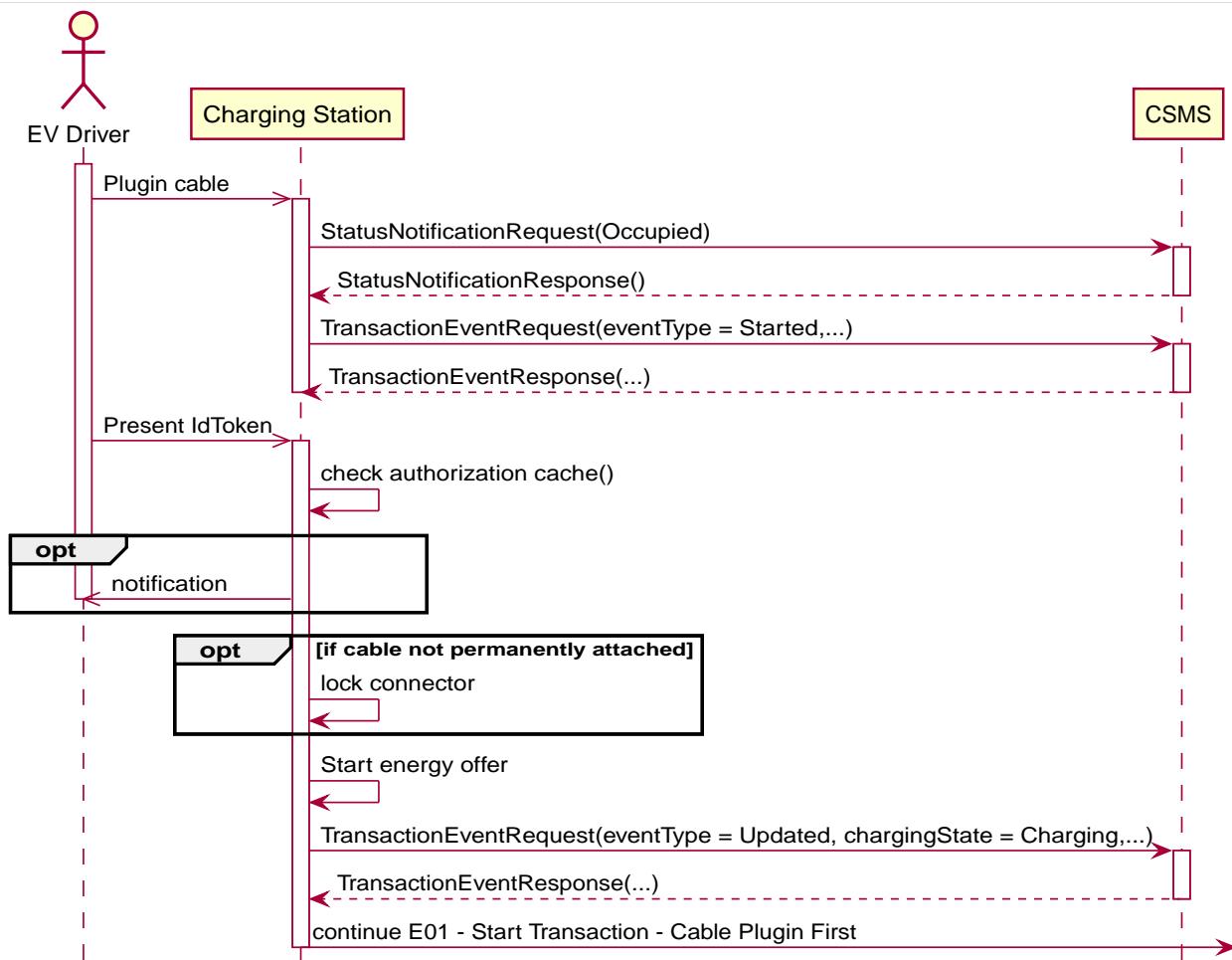


Figure 33. Sequence Diagram: Start Transaction - Cached Id

7	Error handling	When the Charging Station has an <code>IdToken</code> in the Authorization Cache, which is valid in the Authorization Cache, but is no longer valid in the CSMS: The Charging Station will receive the <code>IdTokenInfo</code> in the <code>TransactionEventResponse</code> which contains the newer invalid status. What happens in such a cases depends on the Configuration Variables: <code>MaxEnergyOnInvalidId</code> and <code>StopTxOnInvalidId</code> .
8	Remark(s)	<p>If the Charging Station has implemented an Authorization Cache, then upon receipt of a <code>AuthorizeResponse</code> message the Charging Station updates the Cache entry.</p> <p>For a Cached valid <code>IdToken</code> it is not logical to send <code>AuthorizeRequest</code>. The <code>TransactioneventResponse</code> message also contains the <code>IdToken</code> information. If the <code>IdToken</code> has become no longer valid, the Charging Station will learn this from this <code>TransactioneventResponse</code>. So if the <code>IdToken</code> is no longer valid, the Charging Station might decide to stop the energy offering, and depending on the configuration even stop the transaction.</p> <p>The scenario description and sequence diagram above are based on the Configuration Variable for start transaction being configured as follows:</p> <p><code>TxStartPoint: EVConnected, Authorized, DataSigned, PowerPathClosed, EnergyTransfer</code></p> <p>This use-case is also valid for other configurations, but then the transaction might start/stop at another moment, which might change the sequence in which message are send. For more details see the use case: <code>E01 - Start Transaction options</code>.</p>

C12 - Start Transaction - Cached Id - Requirements

Table 82. C12 - Requirements

ID	Precondition	Requirement definition	Note
C12.FR.02	When an identifier is presented that is stored in the Authorization Cache as <code>Accepted</code> .	The Charging Station SHALL send a <code>TransactionEventRequest</code> with <code>idToken</code> to the CSMS.	

ID	Precondition	Requirement definition	Note
C12.FR.03	C12.FR.02	The CSMS SHALL check the authorization status of the IdToken when processing this TransactionEventRequest .	
C12.FR.04	C12.FR.02 AND The cable is plugged in.	The Charging Station SHALL start the energy offer.	
C12.FR.05	When an identifier is presented that is stored in the Authorization Cache with status other than Accepted, and the Charging Station is online.	The Charging Station SHALL send an AuthorizeRequest to the CSMS.	To check the current state of the identifier.
C12.FR.06	When IdTokenInfo is received for an identifier in the Cache.	The Authorization Cache SHALL be updated using the received IdTokenInfo .	
C12.FR.09	IdTokens that have a groupId equal to MasterPassGroupId	SHALL NOT be allowed to start a transaction.	

2.5. Local Authorization list

C13 - Offline Authorization through Local Authorization List

Table 83. C13 - Offline Authorization through Local Authorization List

No.	Type	Description
1	Name	Offline Authorization through Local Authorization List
2	ID	C13
	Functional block	C. Authorization
3	Objective(s)	To authorize an idToken by using the Local Authorization List while <i>Offline</i> .
4	Description	<p>This use case describes how to authorize an IdToken, when communication with the CSMS is not possible.</p> <p>The Local Authorization List is a list of idTokens that can be synchronized with the CSMS. The list contains the authorization status of a selected set of idTokens as managed by the CSMS.</p>
	Actors	EV Driver, Charging Station
	Scenario description	<ol style="list-style-type: none"> 1. The Charging Station is <i>Offline</i> 2. The EV Driver presents IdToken. 3. The Charging Station checks if the IdToken is known and has status Accepted in the Local Authorization List. 4. The Charging Station start charging.
5	Prerequisite(s)	<p><i>Local Authorization List</i> is available</p> <p><i>Local Authorization List</i> is enabled via LocalAuthListEnabled</p> <p>Charging Station is <i>Offline</i></p> <p>The Id of the EV Driver is in the <i>Local Authorization List</i></p> <p>Id is valid</p>
6	Postcondition(s)	<p>Successful postcondition:</p> <p>The Charging Station accepts tokens on the Local Authorization List when it is offline.</p> <p>Failure postcondition:</p> <p>The Charging Station does not accept tokens on the Local Authorization List when it is offline.</p>

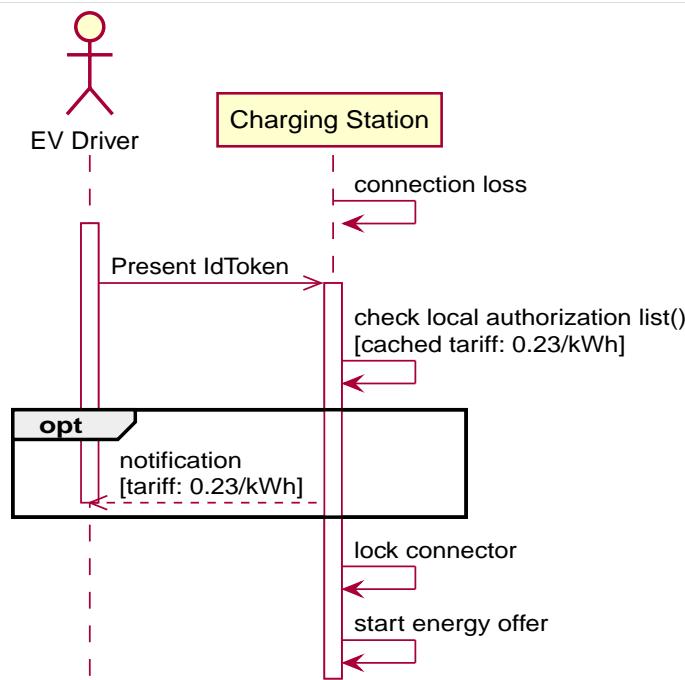


Figure 34. Sequence Diagram: Offline Authorization through Local Authorization List

7	Error handling	n/a
8	Remark(s)	n/a

C13 - Offline Authorization through Local Authorization List - Requirements

Table 84. C13 - Requirements

ID	Precondition	Requirement definition	Note
C13.FR.01		Where both Authorization Cache and Local Authorization List are supported, a Charging Station SHALL treat Local Authorization List entries as having priority over Authorization Cache entries for the same identifiers.	
C13.FR.02	If configuration variable OfflineTxForUnknownIdEnabled is false AND The Charging Station is offline.	Only identifiers that are present in a Local Authorization List that have a status Accepted SHALL be allowed to authorize a transaction.	
C13.FR.03		The Charging Station MAY authorize the IdToken locally without involving the CSMS.	As described in Local Authorization List .
C13.FR.04	If configuration variable OfflineTxForUnknownIdEnabled is true AND The Charging Station is offline.	Any identifier SHALL be allowed to authorize a transaction.	

C14 - Online Authorization through Local Authorization List

Table 85. C14 - Online Authorization through Local Authorization List

No.	Type	Description
1	Name	Online Authorization through Local Authorization List
2	ID	C14
	Functional block	C. Authorization
3	Objective(s)	To authorize an idToken by using the Local Authorization List while Online .
4	Description	This use case describes how to authorize an IdToken via the Local Authorization List while the Charging Station is online. When online the Charging Station can then locally authorize the IdToken, and is not required to send an AuthorizeRequest for a known IdToken.
	Actors	EV Driver, Charging Station

No.	Type	Description
	Scenario description	<p>1. The EV Driver presents IdToken</p> <p>2. The Charging Station checks if the IdToken is known and has status Accepted in the Local Authorization List.</p> <p>3. If the IdToken is not known, or the IdToken is not Accepted the Charging Station sends an AuthorizeRequest</p> <p>4. The Charging Station starts charging.</p>
5	Prerequisite(s)	<p>Local Authorization List is available</p> <p>Local Authorization List is enabled via <code>LocalAuthListEnabled</code></p> <p>The Id of the EV Driver is in the Local Authorization List</p> <p>Id is valid <code>LocalPreAuthorize</code> is set to true</p>
6	Postcondition(s)	<p>Successful postcondition: The Charging Station accepts tokens on the Local Authorization List.</p> <p>Failure postcondition: The Charging Station does not accept tokens on the Local Authorization List.</p>

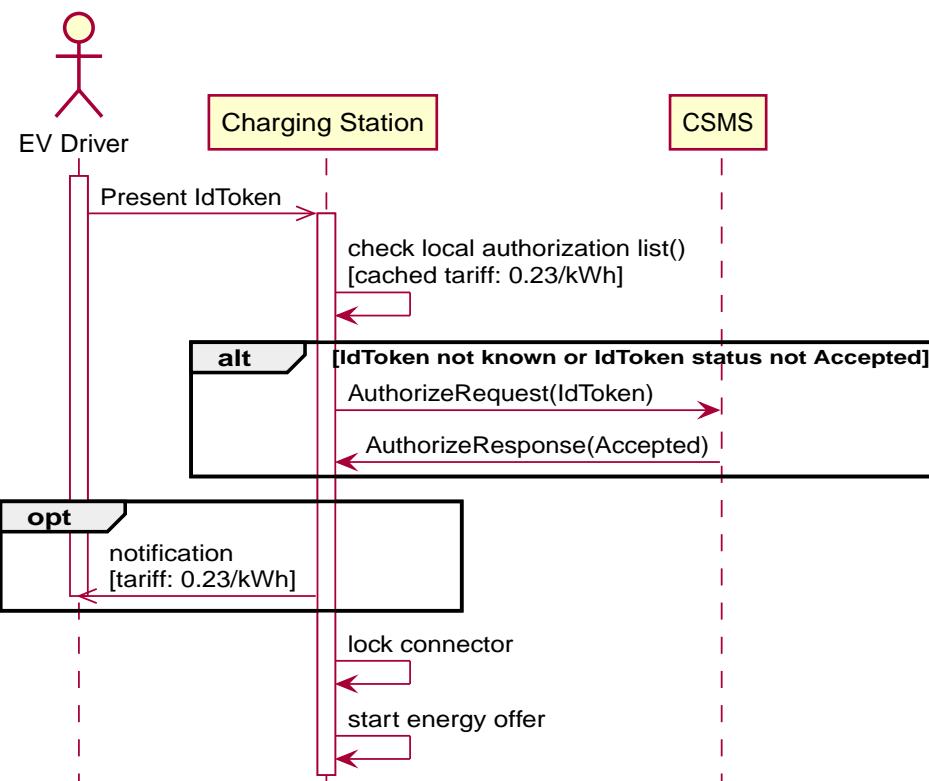


Figure 35. Sequence Diagram: Online Authorization through Local Authorization List

7	Error handling	n/a
8	Remark(s)	n/a

C14 - Online Authorization through Local Authorization List - Requirements

Table 86. C14 - Requirements

ID	Precondition	Requirement definition
C14.FR.01		Where both <code>Authorization Cache</code> and <code>Local Authorization List</code> are supported, a Charging Station SHALL treat <code>Local Authorization List</code> entries as having priority over Authorization Cache entries for the same identifiers.
C14.FR.02	Identifiers presented is in the Local Authorization List with a status Accepted	The Charging Station SHALL start charging without sending an <code>AuthorizeRequest</code> .
C14.FR.03	Identifiers presented is in the Local Authorization List with a status OTHER than Accepted	The Charging Station SHALL send an <code>AuthorizeRequest</code> to try to authorize this IdToken.

2.6. Offline Authorization

C15 - Offline Authorization of unknown Id

Table 87. C15 - Offline Authorization of unknown Id

No.	Type	Description
1	Name	Offline Authorization of unknown Id
2	ID	C15
	Functional block	C. Authorization
	Parent use case	C12 - Start Transaction - Cached Id
3	Objective(s)	To allow automatic authorization of any "unknown" identifiers that cannot be explicitly authorized by Authorization Cache entries.
4	Description	This use case describes the scenario of presented "unknown" identifiers, other than are present in an Authorization Cache or Local Cache entry using OfflineTxForUnknownIdEnabled .
	Actors	Charging Station, EV Driver
	Scenario description	<p>1. The EV Driver wants to start charging the EV and presents the IdToken.</p> <p>2. The Charging Station checks the Authorization Cache, the IdToken is not present in the Authorization Cache.</p> <p>3. The Charging Station checks the Local Authorization List, the IdToken is not present in the Local Authorization List.</p> <p>4. The Charging Station accepts the unknown IdToken if OfflineTxForUnknownIdEnabled is set <i>True</i></p> <p>5. The Charging Station rejects the unknown IdToken if OfflineTxForUnknownIdEnabled is set <i>False</i></p>
	Alternative scenario(s)	C01 - EV Driver Authorization using RFID C02 - Authorization using a start button C03 - Authorization using credit/debit card C04 - Authorization using PIN-code C05 - Authorization for CSMS initiated transactions C06 - Authorization using local id type C07 - Authorization using Contract Certificates C08 - Authorization at EVSE using ISO 15118 External Identification Means (EIM)
5	Prerequisite(s)	The Charging Station is <i>Offline</i> . Unknown IdToken presented (Not in the Authorization Cache and/or Local Authorization List).
6	Postcondition(s)	<p>Successful postcondition: The authorization status in TransactionEventResponse is <i>Accepted</i>.</p> <p>Failure postcondition: The authorization status in TransactionEventResponse is <i>not Accepted</i> when OfflineTxForUnknownIdEnabled is <i>True</i>.</p>

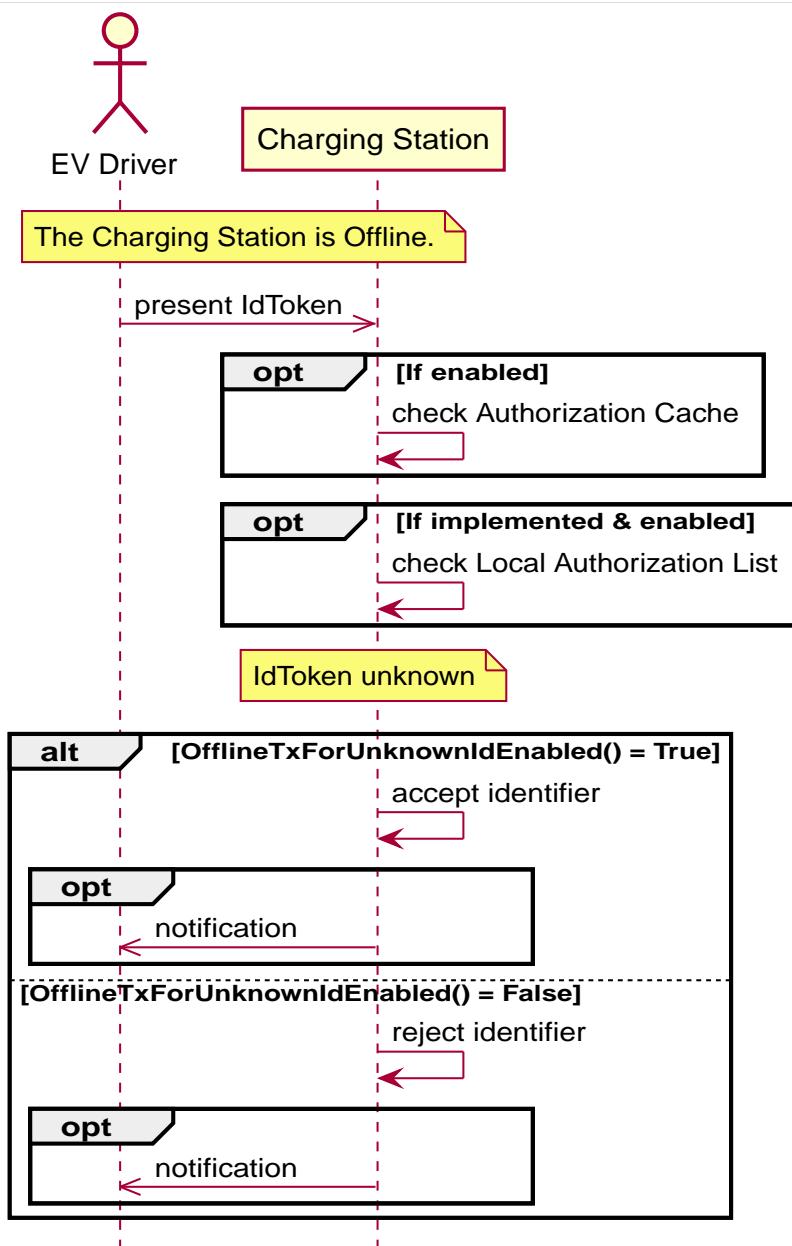


Figure 36. Sequence Diagram: Start Transaction - Unknown Offline Authorization

7	Error handling	n/a
8	Remark(s)	This applies to all types of identifiers, including an eMAID that is presented as part of an ISO 15118 contract certificate.

C15 - Offline Authorization of unknown Id - Requirements

Table 88. C15 - Requirements

ID	Precondition	Requirement definition	Note
C15.FR.01	If the identifier is authorized via <code>OfflineTxForUnknownIdEnabled</code>	The Charging Station SHALL NOT add the token to <code>Authorization Cache</code>	
C15.FR.02	When connection to the CSMS is restored	The Charging Station SHALL send a <code>TransactionEventRequest</code> for any transaction that was authorized <i>offline</i> .	As explained in <code>transaction-related message handling</code>

ID	Precondition	Requirement definition	Note
C15.FR.03	C15.FR.02 AND The authorization status in TransactionEventResponse is not Accepted AND The transaction is still ongoing AND StopTxOnInvalidId is true AND TxStopPoint does NOT contain: (Authorized OR PowerPathClosed OR EnergyTransfer)	The Charging Station SHALL stop the energy transfer and send TransactionEventRequest (eventType = Updated) with triggerReason set to Deauthorized and chargingState set to SuspendedEVSE or preferably to EVConnected.	Since the effect of setting chargingState to SuspendedEVSE or EVConnected both have the same effect of not delivering any energy, the use of SuspendedEVSE is still allowed in this situation in order to avoid breaking existing implementations that adhere to the original requirement. Use of SuspendedEVSE in this situation will become deprecated in the next OCPP release.
C15.FR.04	C15.FR.02 AND The authorization status in TransactionEventResponse is not Accepted AND The transaction is still ongoing AND StopTxOnInvalidId is true AND TxStopPoint does contain: (Authorized OR PowerPathClosed OR EnergyTransfer)	The Charging Station SHALL stop the transaction and send TransactionEventRequest (eventType = Ended) with triggerReason set to Deauthorized and stoppedReason set to DeAuthorized.	
C15.FR.05	C15.FR.04 AND If the Charging Station has the possibility to lock the Charging Cable	The Charging Station SHOULD keep the Charging Cable locked until the owner presents his identifier.	
C15.FR.06	C15.FR.02 AND The authorization status in TransactionEventResponse is not Accepted AND The transaction is still ongoing AND StopTxOnInvalidId is set to false AND MaxEnergyOnInvalidId is not implemented or has been exceeded. TxStopPoint does NOT contain: EnergyTransfer	The Charging Station SHALL stop the energy delivery to the EV immediately and send TransactionEventRequest (eventType = Updated) with triggerReason set to ChargingStateChanged and chargingState set to SuspendedEVSE	
C15.FR.07	C15.FR.02 AND The authorization status in TransactionEventResponse is not Accepted AND The transaction is still ongoing AND StopTxOnInvalidId is set to false AND MaxEnergyOnInvalidId is set and has NOT been exceeded.	Energy delivery to the EV SHALL be allowed until the amount of energy specified in MaxEnergyOnInvalidId has been reached.	
C15.FR.08	When an unknown identifier is presented AND OfflineTxForUnknownIdEnabled is set to true	The Charging Station SHALL accept the presented IdToken.	

2.7. Master Pass

C16 - Stop Transaction with a Master Pass

Table 89. C16 - Stop Transaction with a Master Pass

No.	Type	Description
1	Name	Stop Transaction with a Master Pass
2	ID	C16
	Functional block	C. Authorization
3	Objectives	Enable stopping of transactions by use of a Master Pass (for example for: Law Enforcement officials).
4	Description	This use case covers how somebody with a Master Pass (User) can stop (selected) ongoing transactions, so the cable becomes unlocked. This Master Pass can be configured in: MasterPassGroupId .
	Actors	Charging Station, CSMS, User
	Scenario description	<p>1. The User (Law Enforcement official) presents his IdToken at the Charging Station.</p> <p>2. The Charging Station sends AuthorizeRequest to the CSMS to request authorization.</p> <p>3. Upon receipt of AuthorizeRequest, the CSMS responds with AuthorizeResponse. This response message contains a GroupId that equals the value of the Configuration Variable MasterPassGroupId and the idToken is valid.</p> <p>4a. If the Charging Station has a UI, then the Charging Station "Shows" the Master Pass UI.</p> <p>5a. The user selects which transactions to stop.</p> <p>6a. The Charging Station stops the selected transaction(s) AND sends a TransactionEventRequest (<code>eventType = Ended, stopReason = MasterPass</code>) to the CSMS for every stopped transaction.</p> <p>7a. Upon receipt of TransactionEventRequest the CSMS responds with TransactionEventResponse.</p> <p>4b. If the Charging Station does NOT have a UI, then the Charging Station stops all transactions AND sends a TransactionEventRequest (<code>eventType = Ended, stopReason = MasterPass</code>) to the CSMS for every stopped transaction.</p> <p>5b. Upon receipt of TransactionEventRequest the CSMS responds with TransactionEventResponse.</p>
	Alternative scenario(s)	C01 - EV Driver Authorization
5	Prerequisites	Ongoing Transaction(s) Configuration Variable: MasterPassGroupId set. Users IdToken has groupId equal to the configured MasterPassGroupId .
6	Postcondition(s)	(Selected) transaction(s) stopped.

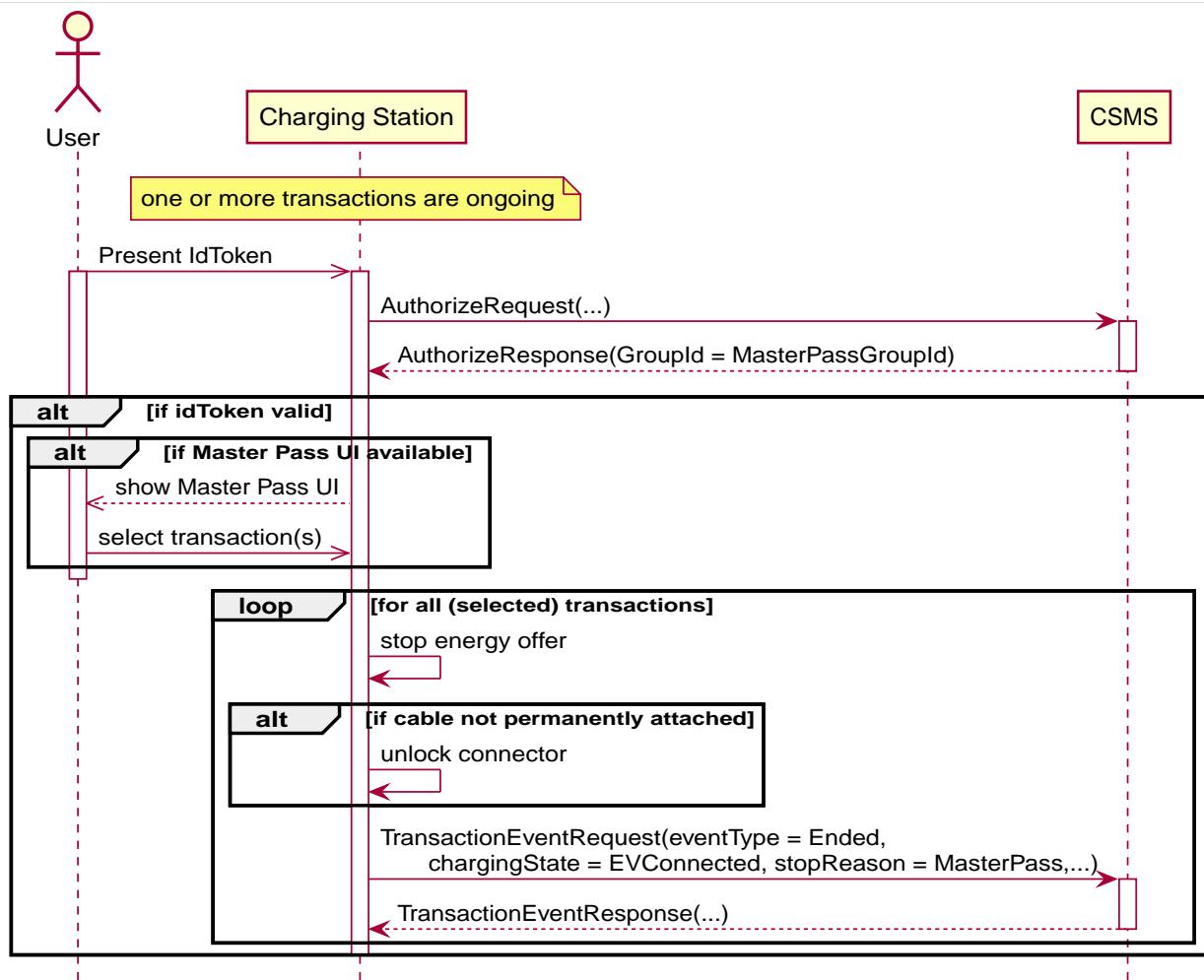


Figure 37. Sequence Diagram: Stop Transaction with a Master Pass

7	Error Handling	When the user does not make a selection before an acceptable timeout, the Charging Station SHALL go back to normal operation.
8	Remarks	<p>The scenario description and sequence diagram above are based on the Configuration Variable for stop transaction being configured as follows.</p> <p>TxStopPoint: Authorized, DataSigned, PowerPathClosed, EnergyTransfer</p> <p>This use-case is also valid for other configurations, but then the transaction might stop at another moment, which might change the sequence in which messages are sent. For more details see the use case: E06 - Stop Transaction options</p>

C16 - Stop Transaction with a Master Pass - Requirements

Table 90. C16 - Stop Transaction with a Master Pass - Requirements

ID	Precondition	Requirement definition
C16.FR.01	User presents an IdToken that has a groupId equal to MasterPassGroupId AND The Charging Station has a UI.	The Charging Station SHALL "show" the Master Pass UI.
C16.FR.02	User presents an IdToken that has a groupId equal to MasterPassGroupId AND the Charging Station does NOT have a UI.	The Charging Station SHALL stop all ongoing transactions.
C16.FR.03	IdTokens that have a groupId equal to MasterPassGroupId	SHALL NOT be allowed to start a transaction.
C16.FR.04	IdTokens that have a groupId equal to MasterPassGroupId present in the Authorization Cache .	The Charging Station MAY also allow authorization of "Master Pass" tokens based on information in the Authorization Cache .
C16.FR.05	IdTokens that have a groupId equal to MasterPassGroupId present in the Local Authorization List .	The Charging Station MAY also allow authorization of "Master Pass" tokens based on information in the Local Authorization List .

D. LocalAuthorizationList Management

1. Introduction

As explained in [C1.4 - Local Authorization List](#), the Local Authorization List is a list of identifiers that can be synchronized with the CSMS. It allows authorization of a user when offline and when online it can be used to reduce authorization response time. This Functional Block is for enabling the CSMS to synchronize the list by either sending a complete list of identifiers to replace the Local Authorization List or by sending a list of changes (add, update, delete) to apply to the Local Authorization List. The operations to support this are [GetLocalListVersion](#) and [SendLocalList](#).

The list contains the authorization status of all (or a selection of) identifiers and the corresponding expiration date. These values may be used to provide more fine grained information to users (e.g. by display message) during local authorization.

2. Use cases & Requirements

D01 - Send Local Authorization List

Table 91. D01 - Send Local Authorization List

No.	Type	Description
1	Name	Send Local Authorization List
2	ID	D01
	Functional block	D. Local Authorization List
3	Objective(s)	To enable the CSMS to send a Local Authorization List which a Charging Station can use for the authorization of idTokens.
4	Description	The CSMS sends a Local Authorization List which a Charging Station can use for the authorization of idTokens. The list MAY be either a full list to replace the current list in the Charging Station or it MAY be a differential list with updates to be applied to the current list in the Charging Station.
	Actors	Charging Station, CSMS
	Scenario description	<ol style="list-style-type: none"> 1. The CSMS sends a SendLocalListRequest to install or update the Local Authorization List. 2. Upon receipt of the SendLocalListRequest the Charging Station responds with a SendLocalListResponse with its status.
5	Prerequisite(s)	Local Authorization List is enabled with Configuration Variable LocalAuthListEnabled .
6	Postcondition(s)	<p>Successful postcondition:</p> <ul style="list-style-type: none"> - A new Local Authorization List is installed on the Charging Station. <p>Failure postcondition:</p> <ul style="list-style-type: none"> - The Local Authorization List on the Charging Station stays as it was. - If the status is <i>Failed</i> or <i>VersionMismatch</i>.

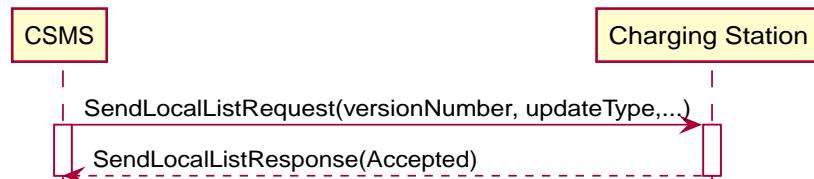


Figure 38. Sequence Diagram: Send Local Authorization List

7	Error handling	If the status is <i>Failed</i> or <i>VersionMismatch</i> and the <code>updateType</code> was Differential, the CSMS will transmit the full Local Authorization List . When this list is too large for one message, it will start by sending an initial list with <code>updateType Full</code> and adding identifiers using <code>updateType Differential</code> until the list is completely sent (the amount of identifiers that can be sent in a single <code>SendLocalListRequest</code> is limited as described in requirement D01.FR.11).
8	Remark(s)	n/a

D01 - Send Local Authorization List - Requirements

Table 92. D01 - Requirements

ID	Precondition	Requirement definition	Note
D01.FR.01		<code>SendLocalListRequest</code> SHALL contain the type of update (<code>updateType</code>) and a version number (<code>versionNumber</code>) that the Charging Station MUST associate with the Local Authorization List after it has been updated.	
D01.FR.02		<code>SendLocalListResponse</code> SHALL indicate whether the Charging Station has accepted the update of the Local Authorization List	

ID	Precondition	Requirement definition	Note
D01.FR.03	If the status in SendLocalListResponse is Failed or VersionMismatch and the updateType was Differential	It is RECOMMENDED that the CSMS sends the full Local Authorization List .	When this list is too large for one message (see D01.FR.11), it shall start by sending an initial list with updateType Full and adding identifiers using updateType Differential until the list is completely sent.
D01.FR.04	If no localAuthorizationList (or an empty one) is given and the updateType is Full.	The Charging Station SHALL remove all IdTokens from the list.	Note, that the version number of the list is still updated to value of versionNumber in the request.
D01.FR.05		Requesting a Differential update without or with empty localAuthorizationList SHALL have no effect on the list.	Note, that the version number of the list is still updated to value of versionNumber in the request.
D01.FR.06		All IdTokens in the Local Authorization List SHALL be unique.	No duplicate values are allowed.
D01.FR.09		The Charging Station SHALL NOT modify the contents of the Authorization List by any other means than upon a the receipt of a SendLocalList message from the CSMS.	
D01.FR.10		The Local Authorization List SHOULD be maintained by the Charging Station in non-volatile memory, and SHOULD be persisted across reboots and power outages.	
D01.FR.11		The size of a single SendLocalListRequest is limited by the Configuration Variables ItemsPerMessageSendLocalList and BytesPerMessageSendLocalList .	
D01.FR.12		A Charging Station that supports Local Authorization List SHALL implement the Configuration Variable: LocalAuthListEntries .	This gives the CSMS a way to known the current amount and maximum possible number of Local Authorization List elements in a Charging Station.
D01.FR.13		The Charging Station indicates whether the Local Authorization List is enabled. This is reported and controlled by the LocalAuthListEnabled Configuration Variable.	
D01.FR.15	If the Charging Station receives a SendLocalListRequest with updateType is Full AND localAuthorizationList is non-empty	The Charging Station SHALL replace its current Local Authorization List with the one in the SendLocalListRequest and set the version number to the value specified in the message	Otherwise, there is no way to sync the initial Charging Station and CSMS lists. When this list is too large for one message (see D01.FR.11), it shall start by sending an initial list with updateType Full and adding identifiers using updateType Differential until the list is completely sent.
D01.FR.16	If the Charging Station receives a SendLocalListRequest with updateType is Differential AND localAuthorizationList contains AuthorizationData elements with idTokenInfo	The Charging Station SHALL update its Local Authorization List with these elements and set the version number to the value specified in the message.	Add them if not yet present, update with new information when already present in the Local Authorization List .

ID	Precondition	Requirement definition	Note
D01.FR.17	If the Charging Station receives a SendLocalListRequest with <i>updateType</i> is Differential AND <i>localAuthorizationList</i> contains AuthorizationData elements without <i>idTokenInfo</i>	The Charging Station SHALL remove these elements from its Local Authorization List and set the version number to the value specified in the message.	
D01.FR.18		<i>versionNumber</i> in a SendLocalListRequest SHALL be greater than 0.	In GetLocalListVersionResponse the <i>versionNumber</i> = 0 has a special meaning: No Local List installed. So the value 0 should never be used.
D01.FR.19	If the Charging Station receives a SendLocalListRequest with <i>updateType</i> = Differential AND <i>versionNumber</i> is less or equal to the version number of its Local Authorization List	The Charging Station SHALL refuse to update its Local Authorization List and SHALL return a SendLocalListResponse with <i>status</i> set to VersionMismatch.	

D02 - Get Local List Version

Table 93. D02 - Get Local List Version

No.	Type	Description
1	Name	Get Local List Version
2	ID	D02
	Functional block	D. Local Authorization List
	Parent use case	D01 - Send Local Authorization List
3	Objective(s)	To support synchronization of Local Authorization List .
4	Description	The CSMS can request a Charging Station for the version number of the Local Authorization List by sending a GetLocalListVersionRequest .
	Actors	Charging Station, CSMS
	Scenario description	1. The CSMS sends a GetLocalListVersionRequest to request this value. 2. Upon receipt of the GetLocalListVersionRequest Charging Station responds with a GetLocalListVersionResponse containing the version number of its Local Authorization List .
5	Prerequisite(s)	
6	Postcondition(s)	The CSMS received the GetLocalListVersionResponse with the Local Authorization List version.

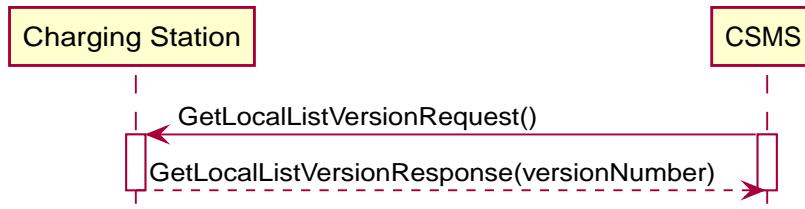


Figure 39. Sequence Diagram: Get Local List Version

7	Error handling	n/a
8	Remark(s)	<p>A <i>versionNumber</i> of 0 (zero) is reserved to indicate that no local authorization list exists, either because it is not enabled or because it has not yet received any update from CSMS and thus does not have a version number to return.</p> <p>In contrast, a local authorization list that was emptied, because CSMS sent a SendLocalListRequest with an empty <i>localAuthorizationList</i>, does have a <i>versionNumber</i> > 0.</p>

D02 - Get Local List Version - Requirements

Table 94. D02 - Requirements

ID	Precondition	Requirement definition
D02.FR.01	LocalAuthListEnabled is true	When Charging Station receives GetLocalListVersionRequest then Charging Station SHALL respond with a GetLocalListVersionResponse containing the version number of its Local Authorization List .
D02.FR.02	LocalAuthListEnabled is true AND the CSMS has not yet sent any update to the Charging Station for Local Authorization List (via SendLocalListRequest)	When Charging Station receives GetLocalListVersionRequest then Charging Station SHALL respond with a GetLocalListVersionResponse with <i>versionNumber</i> is 0 (zero) to indicate that there is no Local Authorization List .
D02.FR.03	LocalAuthListEnabled is not true	When Charging Station receives GetLocalListVersionRequest then Charging Station SHALL respond with a GetLocalListVersionResponse with <i>versionNumber</i> is 0 (zero) to indicate that there is no Local Authorization List .

E. Transactions

1. Introduction

This Functional Block describes the OCPP Transaction related functionalities. Transactions are started/stopped on the Charging Station. Note that at most one transaction can be active on an EVSE at any point in time.

1.1. Flexible transaction start/stop

To support as many business cases as possible, and to prevent sending too many messages when not needed for certain business cases, OCPP 2.0.1 supports flexible configuration of the start and stop of a transaction.

For this the following Configuration Variables are defined:

- `TxStartPoint`
- `TxStopPoint`

These 2 Configuration Variables make it possible to define when a transaction should start: [TransactionEventRequest](#) (eventType = Started) and when a transaction should stop: [TransactionEventRequest](#) (eventType = Ended)

With the introduction in OCPP 2.0.1 of flexible start/stop points of a transaction, it is important to provide a definition of a transaction.

A transaction is the portion of a charging session that is recorded by CSMS. It is a single time frame with a start and stop time. This information can be used by the operator for billing.

It is up to the Charging Station Operator to define the values for TxStartPoint and TxStopPoint (unless these are preset as read-only values in the charging station), but not all combinations make sense.

The following three variants are most common:

- If connection time is billed, then start and stop points should be `EVConnected`.
- If time of use is billed, then the start points should be `EVConnected`, `Authorized` and the stop point `EVConnected`. (Such that upon authorization first, the charger is already seen as 'in use').
- If charging time is billed, then start and stop points should be `PowerPathClosed`. (This starts as soon as charger is ready to provide power and stops when authorization is revoked or vehicle disconnected.) Pauses in between (i.e. `SuspendedEV(SE)`) do not end the transaction. Billing on the amount of energy or power can be done based on the meter values that are collected during the transaction.

WARNING

Certain combinations of start and stop points can lead to a situation where a started transaction is never stopped. For example: when the start point is `ParkingBayOccupancy` and the stop point is `EVConnected`, then a transaction starts when an EV occupies the parking bay, but when the user never connects the EV, but simply drives away, then the transaction will remain open, because `ParkingBayOccupancy` is not configured as a stop point.

1.1.1. Readonly or Read/Write

OCPP 2.0.1 supports 2 options for the transaction start/stop Configuration Variables. They can either be: RW (read-write) or R (read-only).

When a Charging Station supports RW, the CSO can configure the settings. To support all possible settings, the software in the Charging Station has to be more flexible.

With only R, the settings are fixed in firmware, the CSO can read the settings to learn how a Charging Station will behave, but cannot configure it. This makes for a simpler implementation. When the needs of the target market are well known there might be no need to implement the flexible model.

1.1.2. OCPP 1.6 Transaction compatibility

If transactions similar to OCPP 1.6 are wanted, this section describes how the transaction start and stop point should be configured.

In OCPP 1.x the moment a Charging Station should send StartTransaction.req was not defined very precise, generally this was done when the Charging Station was ready to deliver energy: cable is connected and user is authorized.

To support similar transaction start behaviour, the value: *PowerPathClosed* is to be used. (and for completeness, also add: *EnergyTransfer*)

Table 95. The settings for an OCPP 1.6 compatible transaction

Configuration Variable	Values
TxStartPoint	PowerPathClosed
TxStopPoint	EVConnected, Authorized

For stop behavior the *ParkingBayOccupancy* should not be added, OCPP 1.6 did not support this, and in case of a dual socket charging station where somebody is using the 'opposite' connector, the transaction would then be stopped, while the EV could still be charging.

1.2. TransactionId generation

New in OCPP 2.0.1: Transaction IDs are now generated by the Charging Station.

In OCPP 1.x this was done by the CSMS. This had some drawbacks. When a Charging Station was offline it had a transaction which did not have a transactionId.

The TransactionId generated by a Charging Station has to be unique for this Charging Station. During the lifetime of a Charging Station it should never use the same TransactionId twice. Also when the Charging Station is rebooted, power cycled, firmware updated, repaired etc.

OCPP does not specify an algorithm to use, but it is RECOMMENDED to use UUIDs.

1.3. Delivering transaction-related messages

The primary purpose of [TransactionEventRequest](#) messages is to give the CSMS the information that it will later use to bill the transaction. To be sure that the CSMS receives all the necessary information for billing a transaction, OCPP uses two mechanisms: *retrying* and *sequence numbers*.

1.3.1. Retrying

The Charging Station sends [TransactionEventRequest](#) messages to the CSMS System as soon as possible after the events they report on have occurred.

If the Charging Station is offline, or if an error occurs processing the message in transport, the CSMS will be missing billing information. In order to repair the missing information in the CSMS, the Charging Station should retry to deliver this information. When the Charging Station fails to receive a [TransactionEventResponse](#) for a [TransactionEventRequest](#) message within the [message timeout period](#), the Charging Station should follow the retry procedure described in use case [E13 - Transaction-related message not accepted by CSMS](#).

1.3.2. Sequence numbers

When delivery of [TransactionEventRequest](#) messages fails and will be retried later, the result is that [TransactionEventRequest](#) messages may arrive in the CSMS in a different order from the one in which the transaction events occurred at the Charging Station. This in turn would make it difficult for the CSMS to know if it received all [TransactionEventRequest](#) messages about a transaction, which the CSMS may want to know before it starts billing the transaction.

In order to make it possible to know that all [TransactionEventRequest](#) messages about a transaction were received, OCPP uses *sequence numbers* in [TransactionEventRequest](#) messages. For every EVSE, the Charging Station maintains a counter of the number of [TransactionEventRequest](#) messages generated about that EVSE. When generating a new [TransactionEventRequest](#) message, the Charging Station includes the current value of the EVSE's counter in the **seqNo** field of the request, and then increments the counter. With this mechanism, a CSMS can check if it has full information about a transaction by checking that:

- It received a [TransactionEventRequest](#) about the start of the transaction, with a **seqNo** a
- It received a [TransactionEventRequest](#) about the stop of the transaction, with a **seqNo** o greater than a.
- It received a [TransactionEventRequest](#) about the transaction with **seqNo** n for every integer n between a and o

1.3.2.1. Sequence number generation

This section is normative.

When a transaction starts, the Charging Station SHOULD set the seqNo field for the [TransactionEventRequest](#) message to 0. (Implementations with a continuously increasing seqNo are still allowed.)

After each [TransactionEventRequest](#) Charging Station SHALL increase the seqNo by 1.

1.4. Authorization

To simplify the use cases in this functional block, the way an EV Driver is authorized is not part of these use cases. It will simply be called something like: "User authorization successful" or "The EV Driver is authorized by the Charging Station and/or CSMS.". This may be any way of authorizing an EV Driver. See functional block: [C Authorization](#) for all the options and requirements for authorization.

1.5. Clarification for optional fields in TransactionEventRequest

This section is informative.

The TransactionEventRequest contains several optional fields. Some of these fields should only be sent once and should not be repeated in every TransactionEventRequest. The following summary points to the requirements related to these optional fields.

evse

(E01.FR.16) The field *evse* is only provided in the first TransactionEventRequest that occurs after the EV has connected. It is not repeated in all future TransactionEventRequests.

idToken

(E03.FR.01) The field *idToken* is provided once in the first TransactionEventRequest that occurs after the transaction has been authorized.

(E07.FR.02) The field *idToken* is provided once in the TransactionEventRequest that occurs when the authorization of the transaction has been ended.

(C12.FR.02) The above is also the case when authorization was granted because the *idToken* is present in the authorization cache with a *Accepted* status.

(F02.FR.05): The above is also the case when the *idToken* is provided by a RequestStartTransactionRequest.

reservationId

(E03.FR.03/H01.FR.15) The field *reservationId* is only provided in the first TransactionEventRequest that occurs when the transaction has been authorized by the *idToken* for which a reservation existed in the charging station.

(F02.FR.06) The above is also the case when the *idToken* is provided by a RequestStartTransactionRequest.

meterValue

(E02.FR.09) The TransactionEventRequest(*eventType=Started*) must contain the meter values that have been configured in SampledDataCtrlr.TxStartedMeasurands.

(E02.FR.10) A TransactionEventRequest(*eventType=Updated*) must be sent at every interval configured in SampledDataCtrlr.TxUpdatedInterval and contain the meter values that have been configured in SampledDataCtrlr.TxUpdatedMeasurands. If TxUpdatedMeasurands == 0, then no meter values are sent.

(E06.FR.11) The TransactionEventRequest(*eventType=Ended*) must contain the meter values that have been configured in SampledDataCtrlr.TxEndedMeasurands. If SampledDataCtrlr.TxEndedInterval == 0, then only the values taken at start and end of the transaction are included.

transactionInfo.chargingState

(E02.FR.13) Whenever the charging state changes, the Charging Station must send a TransactionEventRequest containing *chargingState*.

A TransactionEventRequest with *triggerReason = ChargingStateChanged* must contain *chargingState*.

transactionInfo.stoppedReason

(C15.FR.04, E05.FR.10, E05.FR.08/09, E07.FR.06) The *stoppedReason* must be provided in the

TransactionEventRequest(*eventType=Ended*), unless the value is *Local*, in which case it may be omitted.

(F03.FR.03, F03.FR.10, F04.FR.03) The above also applies to transactions that are stopped by a RequestStopTransactionRequest, however in this case the *stoppedReason* value must be *Remote*.

transactionInfo.remoteStartId

(C05.FR.03, F01.FR.25, F02.FR.01) The *remoteStartId* must be sent in the next TransactionEventRequest after the RequestStartTransactionRequest with the same *remoteStartId*.

2. Use cases & Requirements

2.1. OCPP transaction mechanism

E01 - Start Transaction options

Table 96. E01 - Start Transaction

No.	Type	Description
1	Name	Start Transaction options
2	ID	E01
	Functional block	E. Transactions
3	Objective(s)	To inform the CSMS that a transaction at the Charging Station has started.
4	Description	This use case describes the different moments a Charging Station can start a transaction (send TransactionEventRequest with <code>eventType = Started</code>), depending on the configuration of the Charging Station.
5	Actors	Charging Station, CSMS, EV Driver
S1	Scenario objective	To start a transaction when a parking bay occupancy detector detects an "EV".
	Scenario description	<p>1. The EV Driver parks his "EV" at a Charging Station with a parking bay occupancy detector, which triggers the detector.</p> <p>2. The Charging Station sends a TransactionEventRequest (<code>eventType = Started</code>) notifying the CSMS about a transaction that has started (even when the driver is not yet known).</p> <p>3. The CSMS responds with a TransactionEventResponse, confirming that the TransactionEventRequest was received.</p>
	Prerequisite(s)	No transaction is ongoing on the EVSE. Configuration Variable: <code>TxStartPoint</code> contains: ParkingBayOccupancy
	Postcondition(s)	<p>Successful postcondition: The transaction is ongoing and the CSMS is <i>Successfully informed</i>.</p> <p>Failure postcondition: The transaction is <i>not</i> ongoing, or The CSMS is <i>not</i> informed.</p>

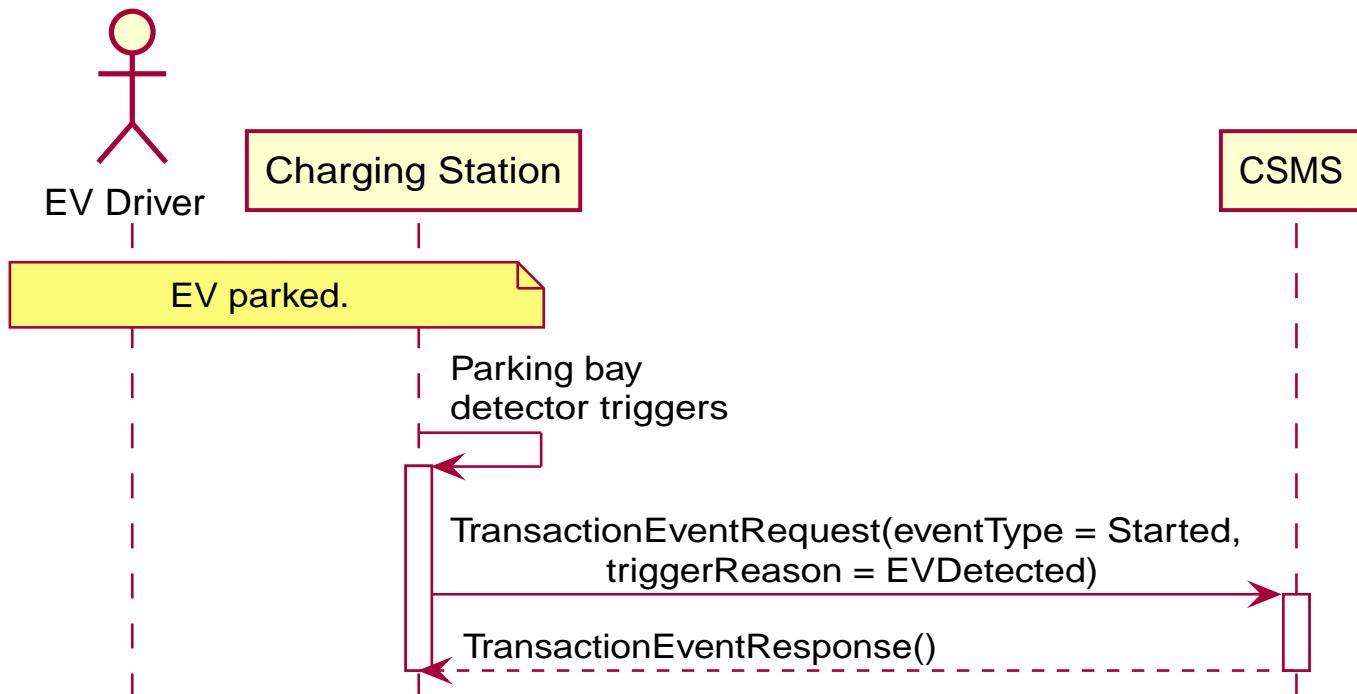


Figure 40. Sequence Diagram: Start Transaction options - ParkingBayOccupancy

S2	Scenario objective	To start a transaction when communication is set up between the Charging Station and an EV (for example: cable plugged in correctly on both sides)
	Scenario description	<p>1. The Charging Station sets up a connection with the EV.</p> <p>2. The Charging Station sends a TransactionEventRequest (<code>eventType = Started</code>) notifying the CSMS about a transaction that has started (even when the driver is not yet known).</p> <p>3. The CSMS responds with a TransactionEventResponse, confirming that the TransactionEventRequest was received.</p>
	Prerequisite(s)	No transaction is ongoing on the EVSE. Configuration Variable: <code>TxStartPoint</code> contains: <code>EVConnected</code> (Not: <code>ParkingBayOccupancy</code>)
	Postcondition(s)	<p>Successful postcondition: The transaction is ongoing and the CSMS is <i>Successfully informed</i>.</p> <p>Failure postcondition: The transaction is <i>not</i> ongoing, or The CSMS is <i>not</i> informed.</p>

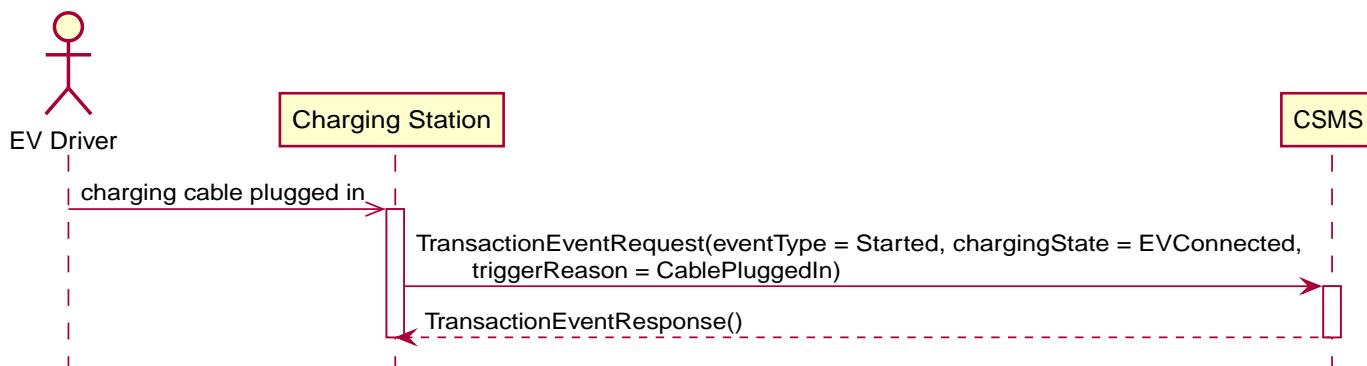


Figure 41. Sequence Diagram: Start Transaction options - EVConnected

S3	Scenario objective	To start a transaction when the EV Driver is authorised to charge.
	Scenario description	<p>1. The EV Driver provides his identification</p> <p>2. The Charging Station validates the provided identification (for example via the Authorization Cache or an AuthorizeRequest).</p> <p>3. The Charging Station sends a TransactionEventRequest (<code>eventType = Started</code>) notifying the CSMS about a transaction that has started.</p> <p>4. The CSMS responds with a TransactionEventResponse, confirming that the TransactionEventRequest was received.</p>
	Prerequisite(s)	No transaction is ongoing on the EVSE. Configuration Variable: <code>TxStartPoint</code> contains: <code>Authorized</code> (Not: <code>ParkingBayOccupancy</code>).
	Postcondition(s)	<p>Successful postcondition: The transaction is ongoing and the CSMS is <i>Successfully informed</i>.</p> <p>Failure postcondition: The transaction is <i>not</i> ongoing, or The CSMS is <i>not</i> informed.</p>

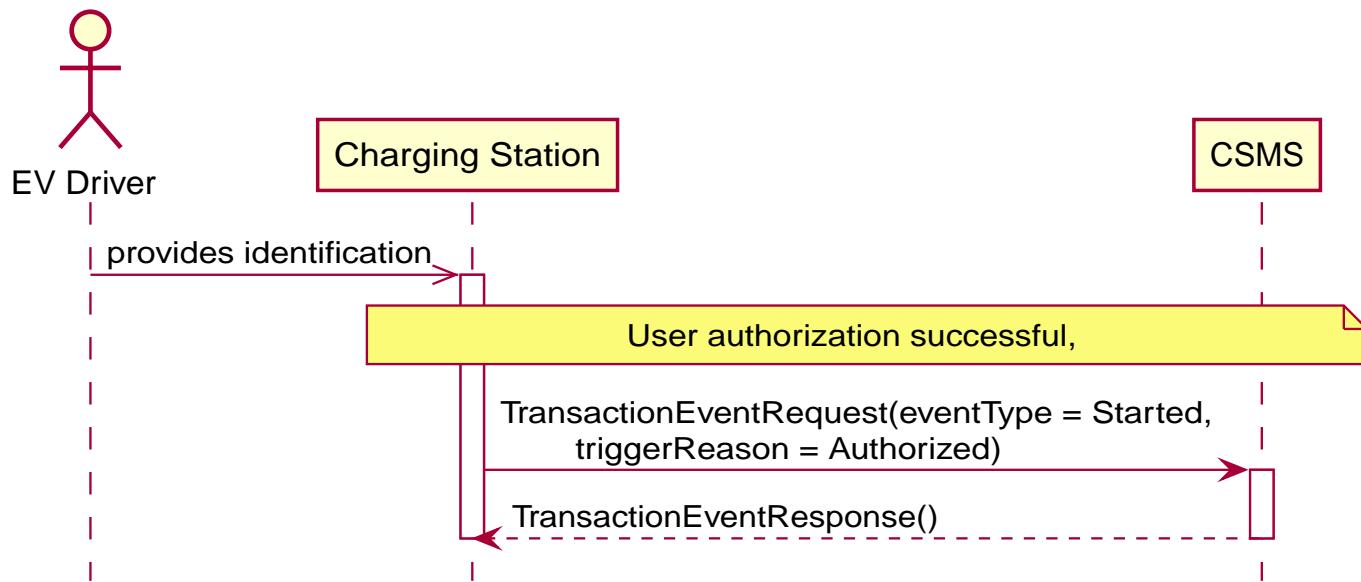


Figure 42. Sequence Diagram: Start Transaction options - Authorized

S4	Scenario objective	To start a transaction when the meter has provided the first signed meter values before starting with charging.
	Scenario description	<ol style="list-style-type: none"> 1. The EV Driver plugs in the cable at the Charging Station and the EV. 2. The Charging Station request the Meter for a signed value. 3. The Meter provides a signed value (this might take some time). 4. The Charging Station sends a TransactionEventRequest (<code>eventType = Started</code>) notifying the CSMS about a transaction that has started. 5. The CSMS responds with a TransactionEventResponse, confirming that the TransactionEventRequest was received.
	Prerequisite(s)	<p>No transaction is ongoing on the EVSE. Configuration Variable: <code>TxStartPoint</code> contains: <code>DataSigned</code> (Not: <code>ParkingBayOccupancy</code>, <code>EVConnected</code> or <code>Authorized</code>). The Charging Station has a meter that can sign measured values Configuration Variable: <code>SampledDataSignReadings</code> set to <code>true</code>.</p>
	Postcondition(s)	<p>Successful postcondition: The transaction is ongoing and the CSMS is <i>Successfully informed</i>.</p> <p>Failure postcondition: The transaction is <i>not</i> ongoing, or The CSMS is <i>not</i> informed.</p>

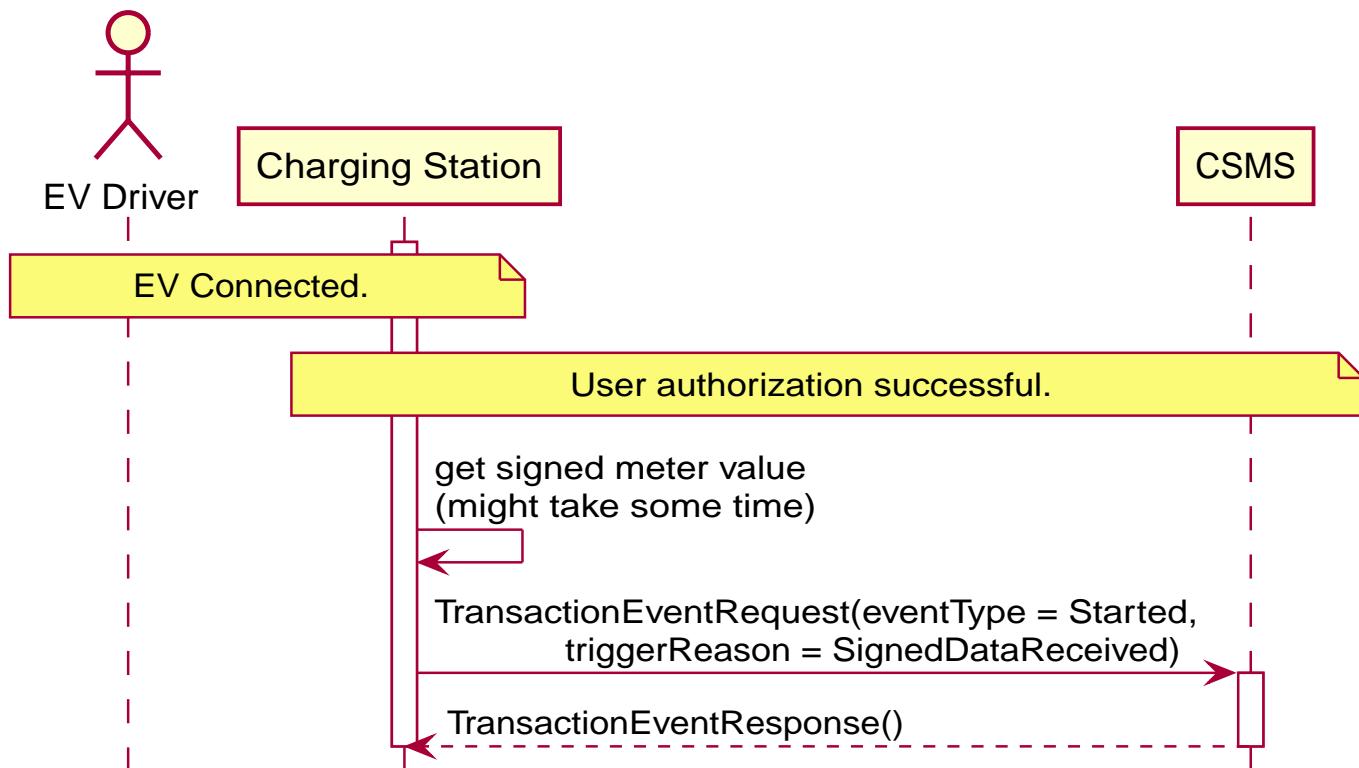


Figure 43. Sequence Diagram: Start Transaction options - DataSigned

S5	Scenario objective	To start a transaction when all preconditions have been met to start charging (authorized and connected), but energy does not yet have to be transferred.
	Scenario description	<ol style="list-style-type: none"> 1. The EV Driver is authorized by the Charging Station and/or CSMS. 2. The Charging Station is connected to the EV. 3. The Charging Station sends a <code>TransactionEventRequest</code> (<code>eventType = Started</code>) notifying the CSMS about a transaction that has started. 4. The CSMS responds with a <code>TransactionEventResponse</code>, confirming that the <code>TransactionEventRequest</code> was received.
	Prerequisite(s)	No transaction is ongoing on the EVSE. Configuration Variable: <code>TxStartPoint</code> contains: <code>PowerPathClosed</code> (Not: <code>ParkingBayOccupancy</code> , <code>EVConnected</code> , <code>Authorized</code> or <code>DataSigned</code>). Charging Cable plugged in.
	Postcondition(s)	<p>Successful postcondition: The transaction is ongoing and the CSMS is <i>Successfully informed</i>.</p> <p>Failure postcondition: The transaction is <i>not</i> ongoing, or The CSMS is <i>not</i> informed.</p>

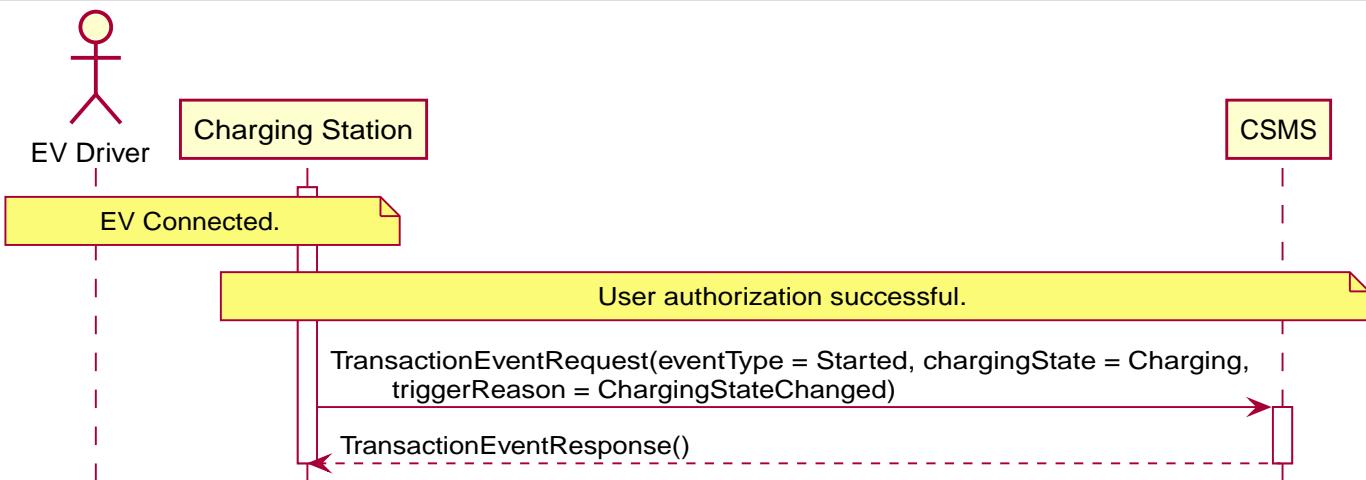


Figure 44. Sequence Diagram: Start Transaction options - PowerPathClosed

S6	Scenario objective	To start a transaction when the energy flow starts.
	Scenario description	<ol style="list-style-type: none"> The EV Driver is authorized by the Charging Station and/or CSMS. The Charging Station closes the power relay. The EV starts charging, energy flow starts. The Charging Station sends a TransactionEventRequest (<code>eventType = Started</code>) notifying the CSMS about a transaction that has started. The CSMS responds with a TransactionEventResponse, confirming that the TransactionEventRequest was received.
	Prerequisite(s)	Configuration Variable: <code>TxStartPoint</code> contains: <code>EnergyTransfer</code> (Not: <code>ParkingBayOccupancy</code> , <code>EVConnected</code> , <code>Authorized</code> , <code>DataSigned</code> or <code>PowerPathClosed</code>).
	Postcondition(s)	Successful postcondition: The transaction is ongoing and the CSMS is <i>Successfully informed</i> . Failure postcondition: The transaction is <i>not</i> ongoing, or The CSMS is <i>not</i> informed.

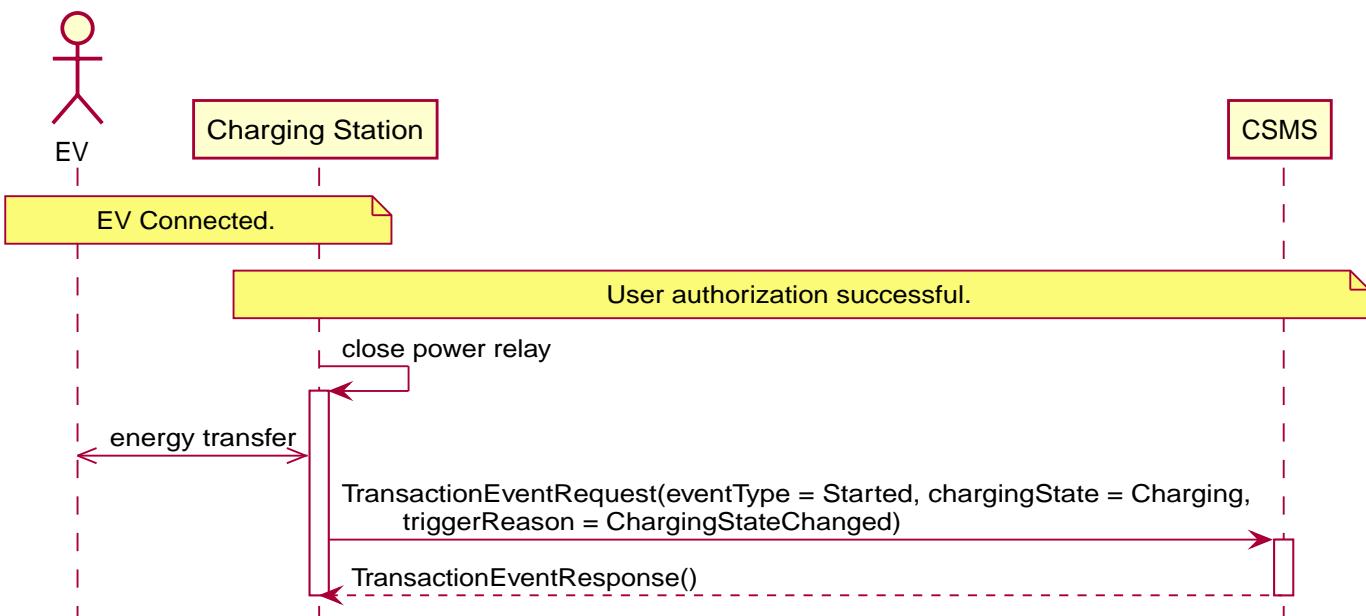


Figure 45. Sequence Diagram: Start Transaction options - EnergyTransfer

7	Error handling	n/a
8	Remark(s)	n/a

E01 - Start Transaction options - Requirements

Table 97. E01 - Requirements

ID	Precondition	Requirement definition
E01.FR.01	<code>TxStartPoint</code> contains: <code>ParkingBayOccupancy</code> AND Parking Bay Detector detects an "EV" AND No transaction has started yet	The Charging Station SHALL start a transaction and send a <code>TransactionEventRequest</code> (<code>eventType = Started</code>) to the CSMS.
E01.FR.02	<code>TxStartPoint</code> contains: <code>EVConnected</code> AND The Charging Station has a connection with the EV AND No transaction has started yet on this EVSE	The Charging Station SHALL start a transaction and send a <code>TransactionEventRequest</code> (<code>eventType = Started</code>) to the CSMS.
E01.FR.03	<code>TxStartPoint</code> contains: <code>Authorized</code> AND The EV Driver is authorized AND No transaction has started yet	The Charging Station SHALL start a transaction and send a <code>TransactionEventRequest</code> (<code>eventType = Started</code>) to the CSMS.
E01.FR.04	<code>TxStartPoint</code> contains: <code>DataSigned</code> AND The Charging Station has a meter that can sign measured values AND Configuration Variable: <code>SampledDataSignReadings</code> set to <code>true</code> . AND The Charging Station has retrieved a signed meter value AND No transaction has started yet	The Charging Station SHALL start a transaction and send a <code>TransactionEventRequest</code> (<code>eventType = Started</code>) to the CSMS.
E01.FR.05	<code>TxStartPoint</code> contains: <code>PowerPathClosed</code> AND The EV Driver is authorized AND The Charging Station has connection with the EV AND No transaction has started yet on this EVSE	The Charging Station SHALL start a transaction and send a <code>TransactionEventRequest</code> (<code>eventType = Started</code>) to the CSMS.
E01.FR.06	<code>TxStartPoint</code> contains: <code>EnergyTransfer</code> AND Energy flow starts AND No transaction has started yet on this EVSE	The Charging Station SHALL start a transaction and send a <code>TransactionEventRequest</code> (<code>eventType = Started</code>) to the CSMS.
E01.FR.07	When a <code>TransactionEventRequest</code> has to be created	The Charging Station SHALL set the message's <code>seqNo</code> field as specified in <code>Sequence Number Generation</code> .
E01.FR.08		The transactionId generated by the Charging Station MUST be unique for each transaction started by that Charging Station, even when the Charging Station is rebooted, repaired, firmware is updated etc, it SHALL ensure that it never generates the same TransactionId twice.

ID	Precondition	Requirement definition
E01.FR.09	When configured to send meter data in the TransactionEventRequest (eventType = Started) , See: Meter Values - Configuration AND EVSE is known at start of transaction	The Charging Station SHALL add the configured measurands to the optional meterValue field with context = Transaction.Begin in the TransactionEventRequest(eventType = Started) sent to the CSMS to provide more details during the transaction.
E01.FR.10	After the EV Driver is authorized for this transaction	The Charging Station SHALL send a TransactionEventRequest that contains IdTokenType information.
E01.FR.11	E01.FR.10	The CSMS SHALL verify the validity of the identifier in TransactionEventRequest .
E01.FR.12	E01.FR.11	The CSMS SHALL send a TransactionEventResponse that includes in idTokenInfo an authorization status value and the groupIdToken if one exists for the idToken .
E01.FR.13	This transaction ends a reservation	The next TransactionEventRequest SHALL contain the reservationId.
E01.FR.14	After TransactionEventRequest(eventType = Started) has been sent for a specific EVSE and Connector	The Charging Station SHALL NOT start another transaction on a different Connector of the same EVSE until this transaction has ended.
E01.FR.15	When sending a TransactionEventRequest	The Charging Station SHALL set the triggerReason to inform the CSMS about what triggered the event. What reason to use is described in the description of TriggerReasonEnumType .
E01.FR.16	After the EV is connected with the Charging Station.	The next TransactionEventRequest SHALL contain evse.id AND evse.connectorId.
E01.FR.17	When configured to send meter data in the TransactionEventRequest (eventType = Started) , See: Meter Values - Configuration AND EVSE is not known at start of transaction	The Charging Station SHALL add the measurands for eventType = Started to the optional meterValue field with context = Transaction.Begin in the TransactionEventRequest(eventType = Updated) that occurs when charging starts.
E01.FR.18	If the charging state changes	The Charging Station SHALL send a TransactionEventRequest including the chargingState element.
E01.FR.19	When EV temporarily suspends the energy transfer	The Charging Station SHOULD send a TransactionEventRequest with chargingState = SuspendedEV
E01.FR.20	E01.FR.19 AND The Charging Station is not able to handle temporary suspension of energy transfer	The Charging Station SHOULD send a TransactionEventRequest with chargingState = EVConnected .

E02 - Start Transaction - Cable Plugin First

Table 98. E02 - Start Transaction - Cable Plugin First

No.	Type	Description
1	Name	Start Transaction - Cable Plugin First
2	ID	E02
	<i>Functional block</i>	E. Transactions
3	Objective(s)	To inform the CSMS that a transaction at the Charging Station has started.
4	Description	The EV Driver begins the interaction with the Charging Station by plugging in the charging cable first. The CSMS is notified about this. Then, when the communication between EV and EVSE is established, the transaction is started and the CSMS is notified of this. The EV starts charging.
	Actors	Charging Station, CSMS, EV Driver

No.	Type	Description
	Scenario description	<p>1. The EV Driver plugs in the cable at the Charging Station.</p> <p>2. The Charging Station sends a StatusNotificationRequest to the CSMS to inform it about a Connector that became <i>Occupied</i>.</p> <p>3. The Charging Station sends a TransactionEventRequest (<code>eventType = Started</code>) notifying the CSMS about a transaction that has started (even when the driver is not yet known.)</p> <p>4. The CSMS responds with a TransactionEventResponse, confirming that the TransactionEventRequest was received.</p> <p>5. The EV Driver is authorized by the Charging Station and/or CSMS.</p> <p>6. The energy offer starts.</p> <p>7. The Charging Station sends a TransactionEventRequest (<code>eventType = Updated</code>) with the authorized idToken information to the CSMS to inform about the charging status and which idToken belongs to the transaction.</p> <p>8. The CSMS responds with a TransactionEventResponse to the Charging Station with the IdTokenInfo.status Accepted.</p> <p>9. During the charging process, the Charging Stations continues to send TransactionEventRequest (<code>Updated</code>) messages for transaction-related notifications.</p>
	Alternative scenario(s)	E02 - Start Transaction - IdToken First E04 - Offline Start Transaction E05 - Start Transaction - Id not Accepted
5	Prerequisite(s)	The Charging Cable is plugged in first.
6	Postcondition(s)	<p>Successful postcondition: The transaction is ongoing and the CSMS is <i>Successfully informed</i>.</p> <p>Failure postcondition: The transaction is <i>not</i> ongoing. or The CSMS is <i>not</i> informed. or Start Transaction - Id not accepted.</p>

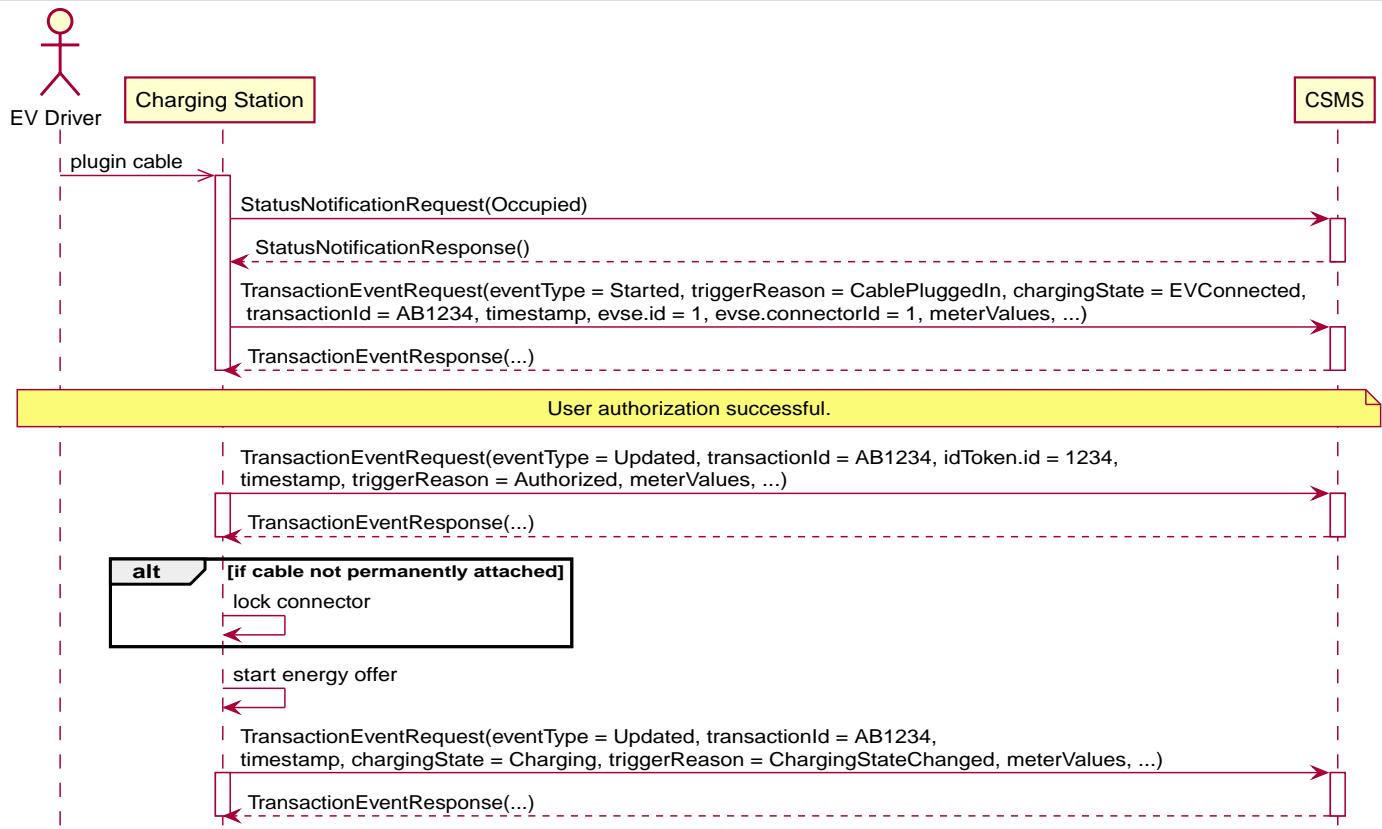


Figure 46. Sequence Diagram: Start Transaction - Cable Plugin First

7	Error handling	Failing to respond with TransactionEventResponse will only cause the Charging Station to try the same message again as specified in E12 - Transaction-related message not accepted by CSMS .
8	Remark(s)	If the Charging Station has implemented an Authorization Cache, then upon receipt of TransactionEventResponse , the Charging Station updates the cache entry. The scenario description and sequence diagram above are based on the Configuration Variable for start & stop transaction being configured as follows: TxStartPoint: EVConnected , Authorized , DataSigned , PowerPathClosed , EnergyTransfer This use-case is also valid for other configurations, but then the transaction might start at another moment, which might change the sequence in which messages are sent. For more details see the use cases: E01 - Start Transaction options and E06 - Stop Transaction options .

E02 - Start Transaction - Cable Plugin First - Requirements

Table 99. E02 - Requirements

ID	Precondition	Requirement definition	Note
E02.FR.01	After the EV Driver is authorized for this transaction.	The next TransactionEventRequest SHALL contain triggerReason: Authorized AND IdTokenType information.	
E02.FR.02	E02.FR.01	The CSMS SHALL send a TransactionEventResponse that includes an authorization status value.	
E02.FR.03	This transaction ends a reservation.	The next TransactionEventRequest SHALL contain the reservationId.	See H. Reservation .
E02.FR.04		The CSMS SHALL verify the validity of the identifier in TransactionEventRequest .	Because the identifier might have been authorized locally by the Charging Station using outdated information.

ID	Precondition	Requirement definition	Note
E02.FR.05	When a cable is plugged in	The Charging Station SHALL send a StatusNotificationRequest with status: <i>Occupied</i>	Alternatively, a NotifyEventRequest message for component (name = 'Connector', evse.id = <x>, evse.connectorId = <y>), variable (name = 'AvailabilityState'), and actualValue = 'Occupied' MAY be sent to signal that Connector <y> of EVSE <x> is now occupied.
E02.FR.06	When a cable is plugged in AND TxStartPoint contains EVConnected	The Charging Station SHALL send a TransactionEventRequest .	
E02.FR.07	When a TransactionEventRequest has to be created	The Charging Station SHALL set the message's seqNo field as specified in Sequence Number Generation .	This enables the CSMS to track the completeness of transaction information.
E02.FR.08		The transactionId generated by the Charging Station MUST be unique for each transaction started by that Charging Station, even when the Charging Station is rebooted, repaired, firmware is updated etc, it SHALL ensure that it never generates the same TransactionId twice.	
E02.FR.09	When configured to send meter data in the TransactionEventRequest (<i>eventType</i> = Started), See: Meter Values - Configuration AND EVSE is known at start of transaction	The Charging Station SHALL add the configured measurands to the optional <i>meterValue</i> field with <i>context</i> = <i>Transaction.Begin</i> in the TransactionEventRequest (<i>eventType</i> = Started) sent to the CSMS to provide more details during the transaction.	
E02.FR.10	When configured to send meter data in the TransactionEventRequest (<i>eventType</i> = Updated), See: Meter Values - Configuration	The Charging Station SHALL add the configured measurands to the optional <i>meterValue</i> field in the TransactionEventRequest (<i>eventType</i> = Updated) sent to the CSMS to provide more details during the transaction.	
E02.FR.11	E02.FR.10 AND Amount of meter data is too much for 1 TransactionEventRequest (<i>eventType</i> = Updated)	The Charging Station MAY split meter data over multiple TransactionEventRequest (<i>eventType</i> = Updated) messages with the same <i>timestamp</i> .	
E02.FR.13	If the charging state changes	The Charging Station SHALL send a TransactionEventRequest including the <i>chargingState</i> element.	
E02.FR.14	SampledDataSignReadings is true	The Charging Station SHALL retrieve signed meter values and put them in the <i>signedMeterValue</i> field of <i>sampledValues</i> .	
E02.FR.15	When sending a TransactionEventRequest	The Charging Station SHALL set the <i>triggerReason</i> to inform the CSMS about what triggered the event. What reason to use is described in the description of TriggerReasonEnumType .	
E02.FR.16	After a transaction has been started	The Charging Station MAY send additional TransactionEventRequest (<i>eventType</i> = Updated) messages during the transaction when a <i>trigger event</i> occurs.	

ID	Precondition	Requirement definition	Note
E02.FR.17	When a transaction-related trigger event occurs, listed in <i>TriggerReasonEnumType</i> AND the transaction is ongoing.	The Charging Station SHALL send a TransactionEventRequest with a triggerReason corresponding to the occurred event.	When two trigger reasons overlap, the more specific one should be used. For example, when a cable is plugged in, triggerReason <i>CablePluggedIn</i> should be used, not <i>ChargingStateChanged</i> . When two events occur at the same time, they need transmitted using two separate TransactionEventRequest messages. This is to prevent information loss, when something goes wrong.
E02.FR.18	When the energy transfer starts AND If the Charging Station is able to report the number of phases used	The Charging Station SHALL provide the number of phases used, using the <i>numberOfPhasesUsed</i> field.	
E02.FR.19	E02.FR.18 AND during the transaction the number of phases used changes	The Charging Station SHALL provide the adjusted number of phases used, using the <i>numberOfPhasesUsed</i> field.	
E02.FR.20	When a transaction has not been authorized before AND the Charging Station authorizes an <i>idToken</i> to start charging	The next TransactionEventRequest from Charging Station SHALL contain the <i>idToken</i> and have <i>triggerReason</i> = Authorized.	If authorization is not successful, then no TransactionEventRequest is sent, because this event has no effect on the running transaction. (For authorization to stop charging, see E07).
E02.FR.21	When configured to send meter data in the TransactionEventRequest (<i>eventType</i> = Started), See: Meter Values - Configuration AND EVSE is not known at start of transaction	The Charging Station SHALL add the measurands for <i>eventType</i> = Started to the optional <i>meterValue</i> field with <i>context</i> = Transaction.Begin in the TransactionEventRequest (<i>eventType</i> = Updated) that occurs when charging starts.	

E03 - Start Transaction - IdToken First

Table 100. E03 - Start Transaction - IdToken First

No.	Type	Description
1	Name	Start Transaction - IdToken First
2	ID	E03
	<i>Functional block</i>	E. Transactions
3	Objective(s)	To enable the EV Driver to start a transaction by first presenting an IdToken at the Charging Station.
4	Description	This use case covers how the EV Driver is first authorized by presenting an IdToken before the cable is plugged in and a transaction starts.
	Actors	Charging Station, CSMS, EV Driver
	<i>Scenario description</i>	<p>1. The EV Driver is authorized by the Charging Station and/or CSMS.</p> <p>2. The Charging Station informs the CSMS that a transaction has started by sending a TransactionEventRequest (<code>eventType = Started</code>).</p> <p>3. The EV Driver plugs in the Charging Cable at the Charging Station.</p> <p>4. The Charging Station sends StatusNotificationRequest to, and receives StatusNotificationResponse from the CSMS.</p> <p>5. The Charging Station informs the CSMS that the EV started charging by sending a TransactionEventRequest (<code>eventType = Updated, chargingState = Charging</code>).</p> <p>6. The CSMS responds with TransactionEventResponse, accepting the transaction.</p>
5	Prerequisite(s)	IdToken is presented prior to plugin cable.
6	Postcondition(s)	<p>Successful postcondition: A transaction is started and the ChargingState is <i>Charging</i></p> <p>Failure postcondition: No transaction is started</p>

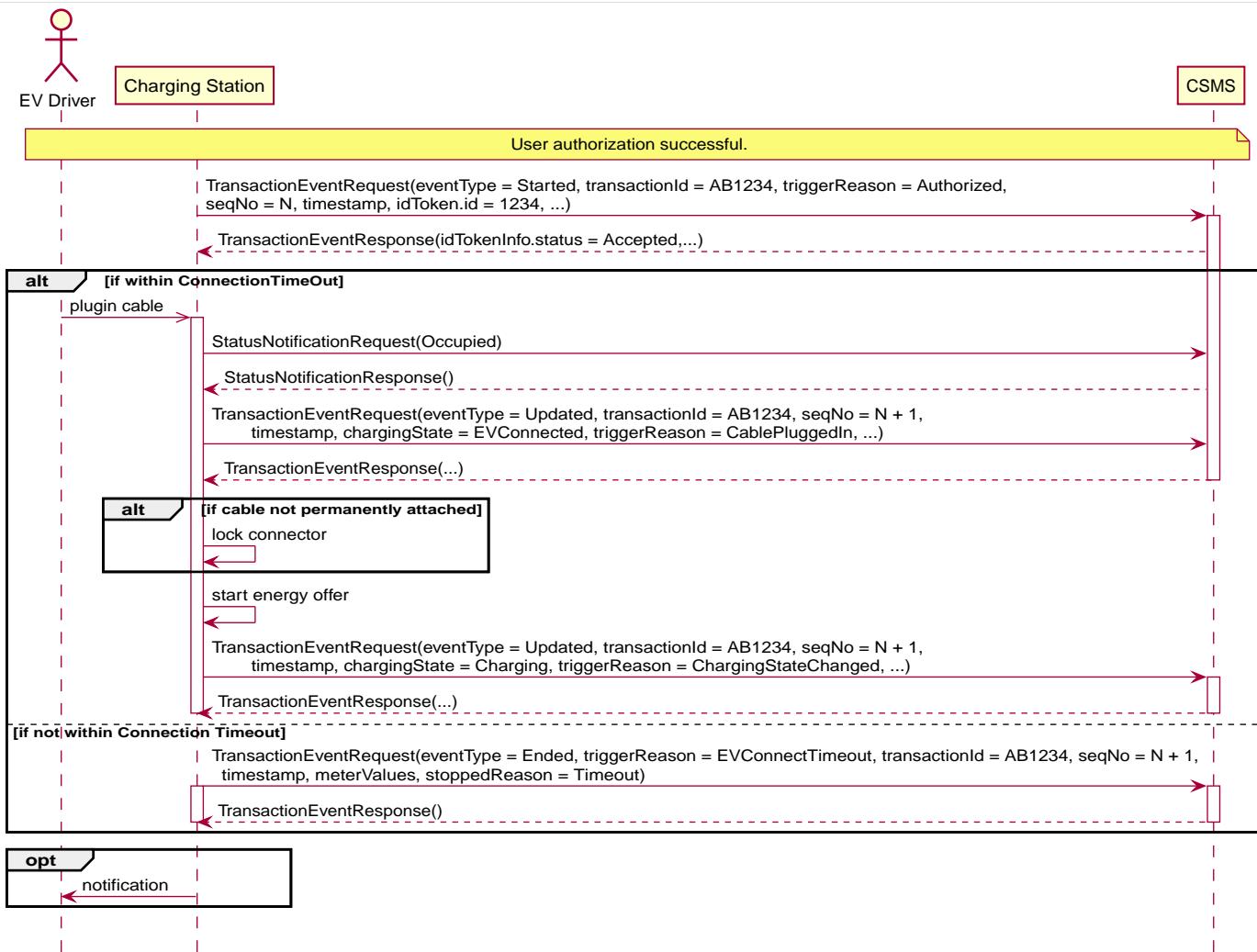


Figure 47. Sequence Diagram: Start Transaction - IdToken First

7	Error handling	n/a
8	Remark(s)	<p>It is likely that the CSMS applies sanity checks to the data contained in TransactionEventRequest messages it received. The outcome of such sanity checks SHOULD NOT ever cause the CSMS to not respond with a TransactionEventResponse. Failing to do so will only cause the Charging Station to try the same message again as specified in E12 - Transaction-related message not accepted by CSMS.</p> <p>The scenario description and sequence diagram above are based on the Configuration Variable for start transaction being configured as follows:</p> <p>TxStartPoint: Authorized, DataSigned, PowerPathClosed, EnergyTransfer</p> <p>This use-case is also valid for other configurations, but then the transaction might start/stop at another moment, which might change the sequence in which message are sent. For more details see the use cases: E01 - Start Transaction options.</p>

E03 - Start Transaction - IdToken First - Requirements

Table 101. E03 - Requirements

ID	Precondition	Requirement definition	Note
E03.FR.01	When the IdToken information is known.	The next TransactionEventRequest SHALL contain IdTokenType information.	
E03.FR.02	E03.FR.01	The CSMS SHALL send a TransactionEventResponse that includes an authorization status.	
E03.FR.03	This transaction ends a reservation for the specific IdToken.	The next TransactionEventRequest SHALL contain the reservationId.	See H. Reservation .

ID	Precondition	Requirement definition	Note
E03.FR.05	When the EV Driver does not plug-in the charging cable before the timeout set by the Configuration Variable: EVConnectionTimeOut AND TxStopPoint does not contain ParkingBayOccupancy	The Charging Station SHOULD end the transaction and send a TransactionEventRequest (eventType = Ended, stoppedReason = Timeout, triggerReason = EVConnectionTimeout) to the CSMS.	This requirement is an additional safety measure to make sure the transaction is ended when the EVConnectionTimeOut is triggered. However it is up to the CSMS to make sure that sensible TxStartPoint / TxStopPoint combinations are configured. E.g. if Authorized is used as TxStartPoint, it should also be used as TxStopPoint.
E03.FR.06	When a TransactionEventRequest has to be created	The Charging Station SHALL set the message's seqNo field as specified in Sequence Number Generation .	This enables the CSMS to track the completeness of transaction information
E03.FR.07	When configured to send meter data in the TransactionEventRequest (eventType = Started), See: Meter Values - Configuration AND EVSE is known at start of transaction	The Charging Station SHALL add the configured measurands to the optional meterValue field with context = Transaction.Begin in the TransactionEventRequest(eventType = Started) sent to the CSMS to provide more details during the transaction.	
E03.FR.08	When configured to send meter data in the TransactionEventRequest (eventType = Updated), See: Meter Values - Configuration	The Charging Station SHALL add the configured measurands to the optional meterValue field in the TransactionEventRequest(eventType = Updated) sent to the CSMS to provide more details during the transaction.	
E03.FR.09	E03.FR.08 AND Amount of meter data is too much for 1 TransactionEventRequest (eventType = Updated)	The Charging Station MAY split meter data over multiple TransactionEventRequest(eventType = Updated) messages with the same timestamp.	
E03.FR.10	SampledDataSignReadings is true	The Charging Station SHALL retrieve signed meter values and put them in the <i>signedMeterValue</i> field of sampledValues.	
E03.FR.11	When configured to send meter data in the TransactionEventRequest (eventType = Started), See: Meter Values - Configuration AND EVSE is not known at start of transaction	The Charging Station SHALL add the measurands for eventType = Started to the optional meterValue field with context = Transaction.Begin in the TransactionEventRequest(eventType = Updated) that occurs when charging starts.	

ID	Precondition	Requirement definition	Note
E03.FR.12	When a transaction-related trigger event occurs, listed in <i>TriggerReasonEnumType</i> AND the transaction is ongoing.	The Charging Station SHALL send a TransactionEventRequest with a triggerReason corresponding to the occurred event.	When two trigger reasons overlap, the more specific one should be used. For example, when a cable is plugged in, triggerReason <i>CablePluggedIn</i> should be used, not <i>ChargingStateChanged</i> . When two events occur at the same time, they need transmitted using two separate TransactionEventRequest messages. This is to prevent information loss, when something goes wrong.
E03.FR.13	When the energy transfer starts AND If the Charging Station is able to report the number of phases used	The Charging Station SHALL provide the number of phases used, using the <i>numberOfPhasesUsed</i> field.	
E03.FR.14	E03.FR.13 AND during the transaction the number of phases used changes	The Charging Station SHALL provide the adjusted number of phases used, using the <i>numberOfPhasesUsed</i> field.	
E03.FR.15	When the EV Driver does not plug-in the charging cable before the timeout set by the Configuration Variable: EVConnectionTimeOut AND TxStopPoint contains <i>ParkingBayOccupancy</i>	The Charging Station SHALL deauthorize the transaction and send a TransactionEventRequest (<i>triggerReason</i> = <i>EVConnectionTimeout</i>) to the CSMS.	Transaction will be ended normally when driver leaves the parking bay.

E04 - Transaction started while Charging Station is offline

Table 102. E04 - Transaction started while Charging Station is offline

No.	Type	Description
1	Name	Transaction started while Charging Station is offline
2	ID	E04
	<i>Functional block</i>	E. Transactions
3	Objective(s)	To enable the EV Driver to start a transaction while the Charging Station is <i>Offline</i> .
4	Description	This use case covers how the Charging Station, while <i>Offline</i> , is able to start a transaction using the Local Authorization List or the Authorization Cache.
	Actors	Charging Station, CSMS, EV Driver
	<i>Scenario description</i>	<ol style="list-style-type: none"> 1. The transaction starts. 2. The TransactionEventRequest (eventType = Started) is stored/queued by the Charging Station. 3. The connection between Charging Station and CSMS is restored. 4. The Charging Station starts to send queued messages 5. The stored TransactionEventRequest is sent, notifying the CSMS about the transaction that was started.
	<i>Alternative scenario(s)</i>	E10 - Connection Loss During Transaction
5	Prerequisite(s)	The Charging Station is <i>Offline</i> . The EV Driver is offline/locally authorized by the Charging Station.
6	Postcondition(s)	<p>Successful postcondition: The TransactionEventRequest has been responded to by the CSMS AND has been removed from the queue of the Charging Station.</p> <p>Failure postcondition: The TransactionEventRequest was NOT responded to by the CSMS AND remains in the queue of the Charging Station.</p>

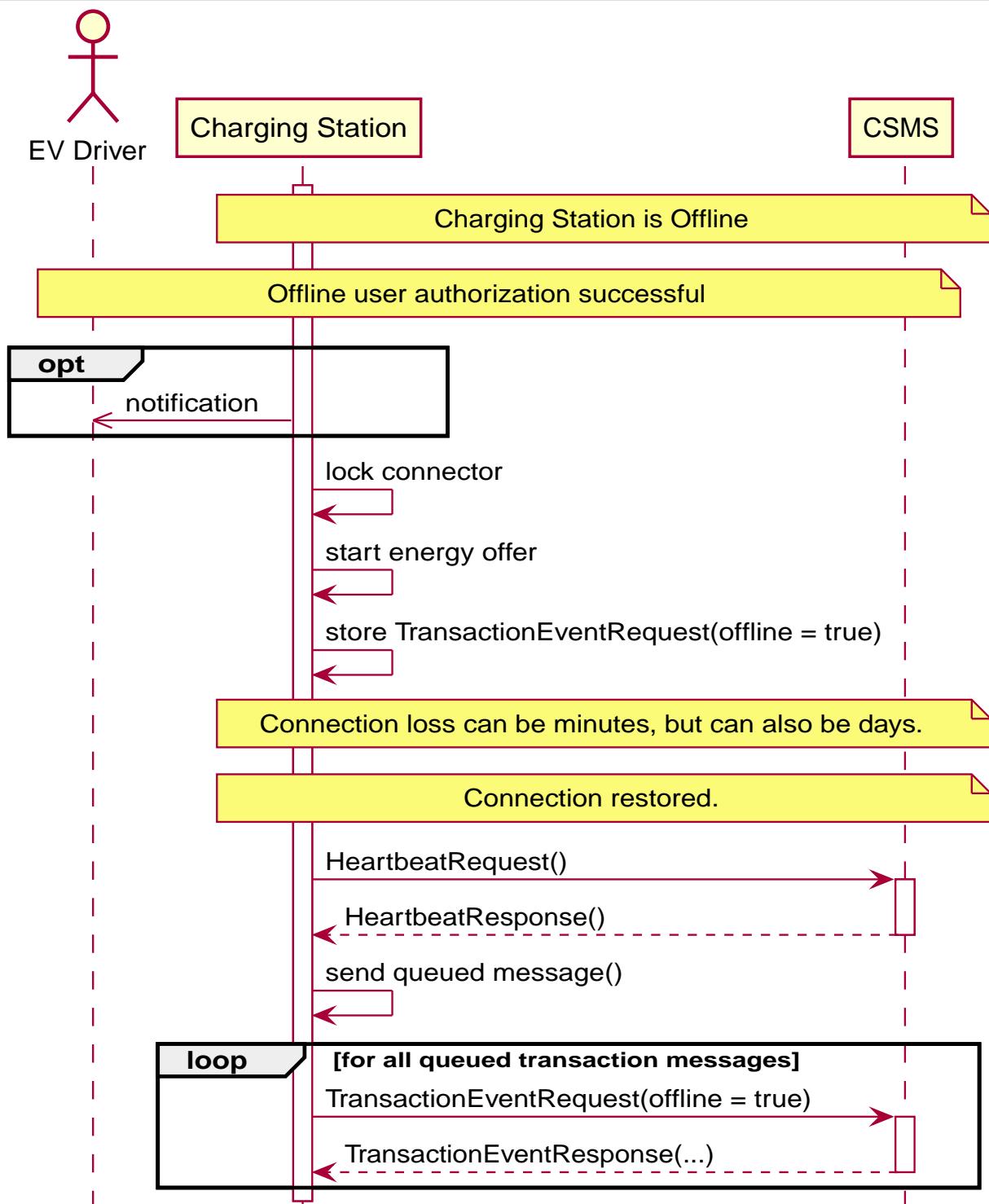


Figure 48. Sequence Diagram: Transaction started while Charging Station is offline

7	Error handling	n/a
8	Remark(s)	<p>The scenario description and sequence diagram above are based on the Configuration Variable for start transaction being configured as follows:</p> <p>TxStartPoint: <code>Authorized</code>, <code>DataSigned</code>, <code>PowerPathClosed</code>, <code>EnergyTransfer</code></p> <p>This use-case is also valid for other configurations, but then the transaction might start/stop at another moment, which might change the sequence in which message are sent. For more details see the use cases: E01 - Start Transaction options.</p>

E04 - Transaction started while Charging Station is offline - Requirements

Table 103. E04 - Requirements

ID	Precondition	Requirement definition	Note
E04.FR.01	When Offline.	The Charging Station MUST queue any TransactionEventRequest messages.	
E04.FR.02	After the connection is restored.	The Charging Station MUST send queued TransactionEventRequest messages.	
E04.FR.03	E04.FR.02	The flag: "offline" SHALL be set to TRUE for any TransactionEventRequest that occurred while the Charging Station was offline.	
E04.FR.04	When a TransactionEventRequest has to be created	The Charging Station SHALL set the message's seqNo field as specified in Sequence Number Generation .	This enables the CSMS to track the completeness of transaction information
E04.FR.05	When configured to send meter data in the TransactionEventRequest (<code>eventType = Started</code>), See: Meter Values - Configuration AND EVSE is known at start of transaction	The Charging Station SHALL add the configured measurands to the optional <code>meterValue</code> field with <code>context = Transaction.Begin</code> in the TransactionEventRequest (<code>eventType = Started</code>) sent to the CSMS to provide more details during the transaction.	
E04.FR.06	When configured to send meter data in the TransactionEventRequest (<code>eventType = Updated</code>), See: Meter Values - Configuration	The Charging Station SHALL add the configured measurands to the optional <code>meterValue</code> field in the TransactionEventRequest (<code>eventType = Updated</code>) sent to the CSMS to provide more details during the transaction.	
E04.FR.07	E04.FR.06 AND <i>Offline</i> AND The Charging Station is running low on memory	The Charging Station MAY drop TransactionEventRequest (<code>eventType = Updated</code>) messages.	
E04.FR.08	E04.FR.07	When dropping TransactionEventRequest (<code>eventType = Updated</code>) messages, the Charging Station SHALL drop intermediate messages first (1st message, 3th message, 5th message etc.), not start dropping messages from the start or stop adding messages to the queue.	
E04.FR.09	E04.FR.06 AND Amount of meter data is too much for 1 TransactionEventRequest (<code>eventType = Updated</code>)	The Charging Station MAY split meter data over multiple TransactionEventRequest (<code>eventType = Updated</code>) messages with the same <code>timestamp</code> .	
E04.FR.10	SampledDataSignReadings is true	The Charging Station SHALL retrieve signed meter values and put them in the <code>signedMeterValue</code> field of <code>sampledValues</code> .	

ID	Precondition	Requirement definition	Note
E04.FR.11	When configured to send meter data in the TransactionEventRequest (<i>eventType</i> = Started), See: Meter Values - Configuration AND EVSE is not known at start of transaction	The Charging Station SHALL add the measurands for <i>eventType</i> = Started to the optional <i>meterValue</i> field with <i>context</i> = Transaction.Begin in the TransactionEventRequest (<i>eventType</i> = Updated) that occurs when charging starts.	

E05 - Start Transaction - Id not Accepted

Table 104. E05 - Start Transaction - Id not Accepted

No.	Type	Description
1	Name	Start Transaction - Id not Accepted
2	ID	E05
	Functional block	E. Transactions
3	Objective(s)	To enable the Charging Station to suspend a transaction when the IdToken has an AuthorizationStatus that does not allow charging.
4	Description	This use case covers how the Charging Station wants to start a transaction while the IdToken is not accepted by the CSMS. Because the identifier might have been authorized locally by the Charging Station using outdated information, the CSMS has to validate the IdTokenType in every TransactionEventRequest message it receives that contains an IdTokenType . When receiving a TransactionEventResponse message with idTokenInfo field status is not Accepted , the Charging Station should stop the energy delivery to the EV.
	Actors	Charging Station, CSMS
	Scenario description	<ol style="list-style-type: none"> 1. The Charging Station sends TransactionEventRequest (eventType = Started) that contains the IdToken provided by the EV Driver. 2. The CSMS responds with TransactionEventResponse, with an AuthorizationStatus that does not allow charging. 3. The Charging Station suspends the energy offer. (Taking into account: MaxEnergyOnInvalidId, if supported) 4. The Charging Station sends TransactionEventRequest (eventType = Updated) with trigger Deauthorized and the chargingState SuspendedEVSE and receives TransactionEventResponse from the CSMS.
5	Prerequisite(s)	The EV Driver is offline/locally authorized by the Charging Station. The IdToken is not allowed to charge by the CSMS.
6	Postcondition(s)	<p>Successful postcondition: The transaction is kept ongoing, and the cable remains locked, but no energy is delivered.</p> <p>Failure postcondition: n/a</p>

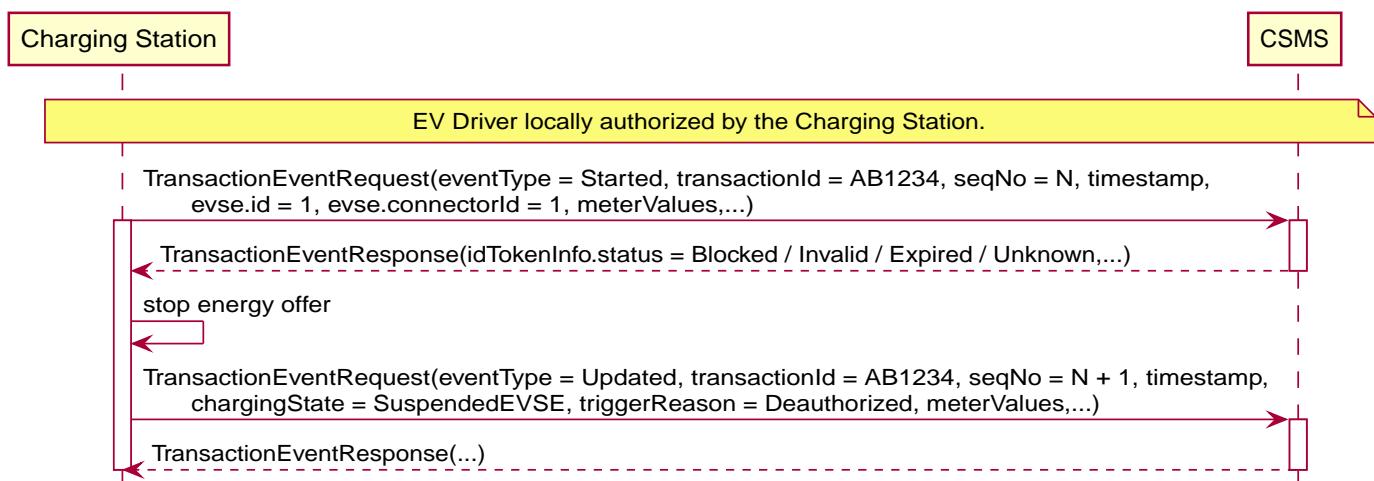


Figure 49. Sequence Diagram: Start Transaction - Id not Accepted

7	Error handling	n/a
---	----------------	-----

8	Remark(s)	The scenario description and sequence diagram above are based on the Configuration Variable for start & stop transaction being configured as follows: TxStartPoint : Authorized, DataSigned, PowerPathClosed, EnergyTransfer TxStopPoint : ParkingBayOccupancy, EVConnected This use-case is also valid for other configurations, but then the transaction might start/stop at another moment, which might change the sequence in which message are sent. For more details see the use cases: E01 - Start Transaction options and E06 - Stop Transaction options .
---	------------------	---

E05 - Start Transaction - Id not Accepted - Requirements

Table 105. E05 - Requirements

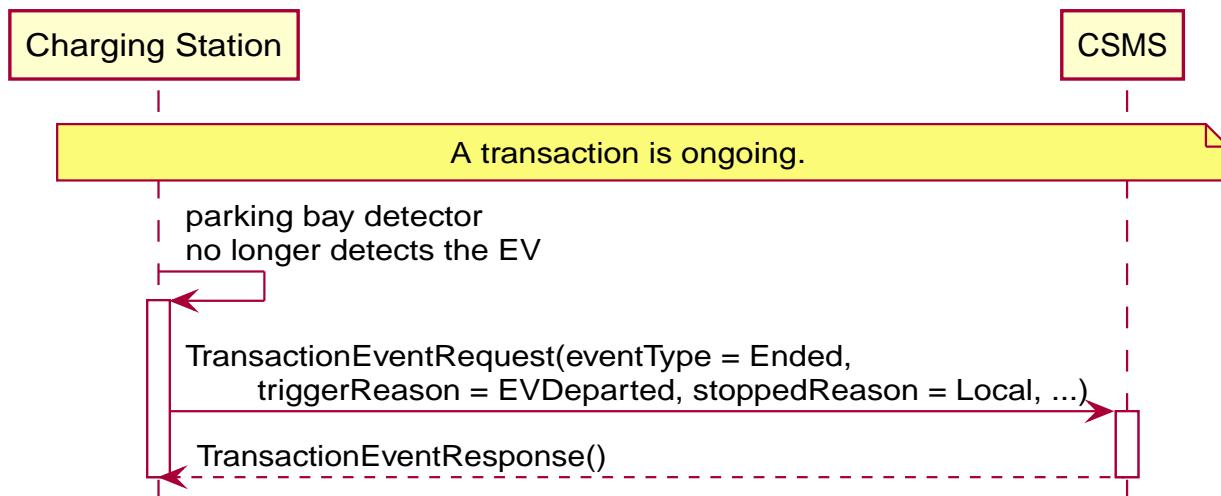
ID	Precondition	Requirement definition	Note
E05.FR.01		The CSMS MUST verify validity of the identifier in the TransactionEventRequest message.	The identifier might have been authorized locally by the Charging Station using outdated information. The identifier, for instance, may have been blocked since it was added to the Charging Station's Authorization Cache.
E05.FR.02	E05.FR.01 AND The authorization status in TransactionEventResponse is not Accepted AND The transaction is still ongoing AND StopTxOnInvalidId is set to <i>false</i> AND MaxEnergyOnInvalidId is not implemented or has been exceeded. TxStopPoint does NOT contain: EnergyTransfer	The Charging Station SHALL stop the energy delivery to the EV immediately and send TransactionEventRequest (eventType = Updated) with <i>triggerReason</i> set to <i>ChargingStateChanged</i> and <i>chargingState</i> set to <i>SuspendedEVSE</i>	The transaction is not deauthorized, but transfer of energy stops, since MaxEnergyOnInvalidId has been exceeded or is not set. If TxStopPoint contains EnergyTransfer then this would have ended the transaction.
E05.FR.03	E05.FR.01 AND The authorization status in TransactionEventResponse is not Accepted AND The transaction is still ongoing AND StopTxOnInvalidId is set to <i>false</i> AND MaxEnergyOnInvalidId is set and has NOT been exceeded.	Energy delivery to the EV SHALL be allowed until the amount of energy specified in MaxEnergyOnInvalidId has been reached.	
E05.FR.04	When a TransactionEventRequest has to be created	The Charging Station SHALL set the message's seqNo field as specified in Sequence Number Generation .	This enables the CSMS to track the completeness of transaction information.
E05.FR.05	When configured to send meter data in the TransactionEventRequest (eventType = Started) , See: Meter Values - Configuration AND EVSE is known at start of transaction	The Charging Station SHALL add the configured measurands to the optional <i>meterValue</i> field with <i>context = Transaction.Begin</i> in the TransactionEventRequest(eventType = Started) sent to the CSMS to provide more details during the transaction.	
E05.FR.06	SampledDataSignReadings is <i>true</i>	The Charging Station SHALL retrieve signed meter values and put them in the <i>signedMeterValue</i> field of sampledValues .	
E05.FR.08	When configured to send meter data in the TransactionEventRequest (eventType = Started) , See: Meter Values - Configuration AND EVSE is not known at start of transaction	The Charging Station SHALL add the measurands for <i>eventType = Started</i> to the optional <i>meterValue</i> field with <i>context = Transaction.Begin</i> in the TransactionEventRequest(eventType = Updated) that occurs when charging starts.	

ID	Precondition	Requirement definition	Note
E05.FR.09	E05.FR.01 AND The authorization status in TransactionEventResponse is not Accepted AND The transaction is still ongoing AND StopTxOnInvalidId is true AND TxStopPoint does NOT contain: (Authorized OR PowerPathClosed OR EnergyTransfer)	The Charging Station SHALL stop the energy transfer and send TransactionEventRequest (eventType = Updated) with triggerReason set to Deauthorized and in the same or next TransactionEventRequest report chargingState set preferably to EVConnected , or alternatively to SuspendedEVSE .	If the physical change of charging state in the Charging Station occurs a few seconds or milliseconds later than the trigger Deauthorized , then the chargingState change may be reported separately as a triggerReason = ChargingStateChanged . Use of charging state SuspendedEVSE that is not followed by EVConnected in this situation will become deprecated in the next OCPP release.
E05.FR.10	E05.FR.01 AND The authorization status in TransactionEventResponse is not Accepted AND The transaction is still ongoing AND StopTxOnInvalidId is true AND TxStopPoint does contain: (Authorized OR PowerPathClosed OR EnergyTransfer)	The Charging Station SHALL stop the transaction and send TransactionEventRequest (eventType = Ended) with triggerReason set to Deauthorized and stoppedReason set to DeAuthorized .	
E05.FR.11	E05.FR.10 AND If the Charging Station has the possibility to lock the Charging Cable	The Charging Station SHOULD keep the Charging Cable locked until the owner presents his identifier.	

E06 - Stop Transaction options

Table 106. E06 - Stop Transaction

No.	Type	Description
1	Name	Stop Transaction options
2	ID	E06
	Functional block	E. Transactions
3	Objective(s)	To inform the CSMS that a transaction at the Charging Station has stopped.
4	Description	This use case describes the different moment a Charging Station can stop a transaction (send TransactionEventRequest (<code>eventType = Ended</code>)), depending on the configuration of the Charging Station.
5	Actors	Charging Station, CSMS, EV Driver
S1	Scenario objective	Stop a transaction when a parking bay occupancy no longer detector detects the EV.
	Scenario description	<ol style="list-style-type: none"> 1. The Charging Stations parking bay occupancy detector stops detecting the EV. 2. The Charging Station sends a TransactionEventRequest (<code>eventType = Ended</code>) notifying the CSMS about a transaction that has ended. 3. The CSMS responds with a TransactionEventResponse, confirming that the TransactionEventRequest was received.
	Prerequisite(s)	A transaction is ongoing. Configuration Variable: TxStopPoint contains: ParkingBayOccupancy
	Postcondition(s)	<p>Successful postcondition: The transaction is ended and the CSMS is <i>Successfully informed</i>.</p> <p>Failure postcondition: The transaction is still ongoing. or The CSMS is <i>not informed</i>.</p>

Figure 50. Sequence Diagram: Stop Transaction options - [ParkingBayOccupancy](#)

S2	Scenario objective	Stop a transaction when communication between the Charging Station and the EV is lost. (for example: cable unplugged)
	Scenario description	<ol style="list-style-type: none"> 1. Communication between Charging Station and the EV is lost (Charging cable is unplugged). 2. If charging cable unplugged on the Charging Station side: send StatusNotificationRequest to the CSMS to inform it about a Connector that became Available. 3. The Charging Station sends a TransactionEventRequest (<code>eventType = Ended</code>) notifying the CSMS about a transaction that has ended. 4. The CSMS responds with a TransactionEventResponse, confirming that the TransactionEventRequest was received.
	Prerequisite(s)	A transaction is ongoing. Configuration Variable: TxStopPoint contains: EVConnected

S2	Scenario objective	Stop a transaction when communication between the Charging Station and the EV is lost. (for example: cable unplugged)
	Postcondition(s)	<p>Successful postcondition: The transaction is ended and the CSMS is <i>Successfully informed</i>.</p> <p>Failure postcondition: The transaction is still ongoing. or The CSMS is <i>not informed</i>.</p>

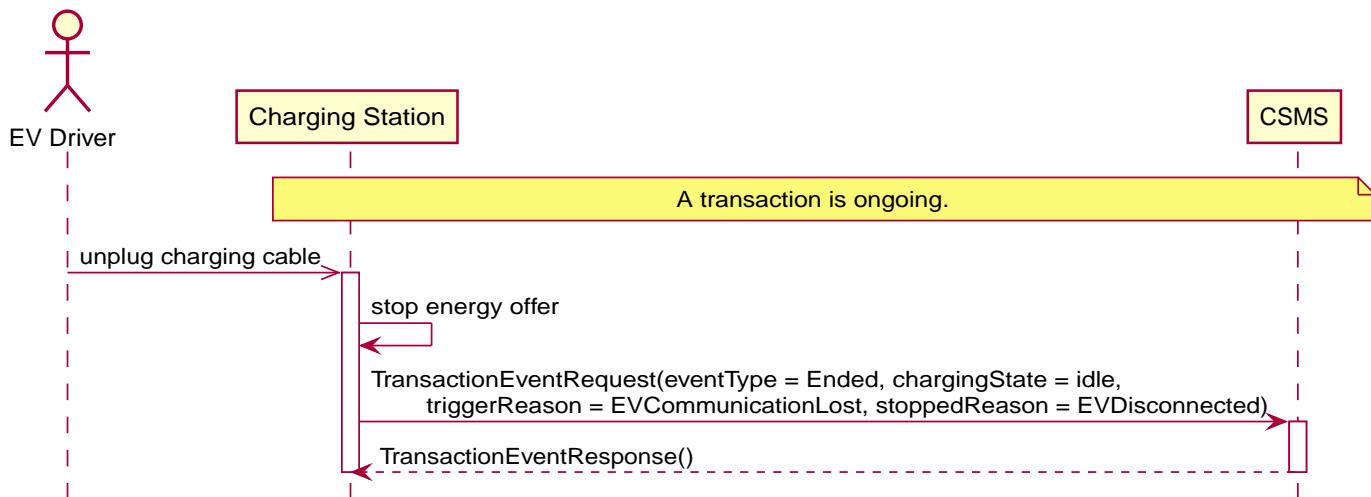


Figure 51. Sequence Diagram: Stop Transaction options - EVConnected

S3	<i>Scenario objective</i>	Stop a transaction when the driver is no longer authorized.
	<i>Scenario description</i>	<p>1. The Charging Station sends a TransactionEventRequest to the CSMS.</p> <p>2. An invalid IdToken is received in a TransactionEventResponse.</p> <p>3. The Charging Station sends a TransactionEventRequest (eventType = Ended) notifying the CSMS about a transaction that has ended.</p> <p>4. The CSMS responds with a TransactionEventResponse, confirming that the TransactionEventRequest was received.</p>
	Prerequisite(s)	<p>A transaction is ongoing.</p> <p>Configuration Variable: TxStopPoint contains: Authorized</p>
	Postcondition(s)	<p>Successful postcondition: The transaction is ended and the CSMS is <i>Successfully informed</i>.</p> <p>Failure postcondition: The transaction is still ongoing. or The CSMS is <i>not informed</i>.</p>

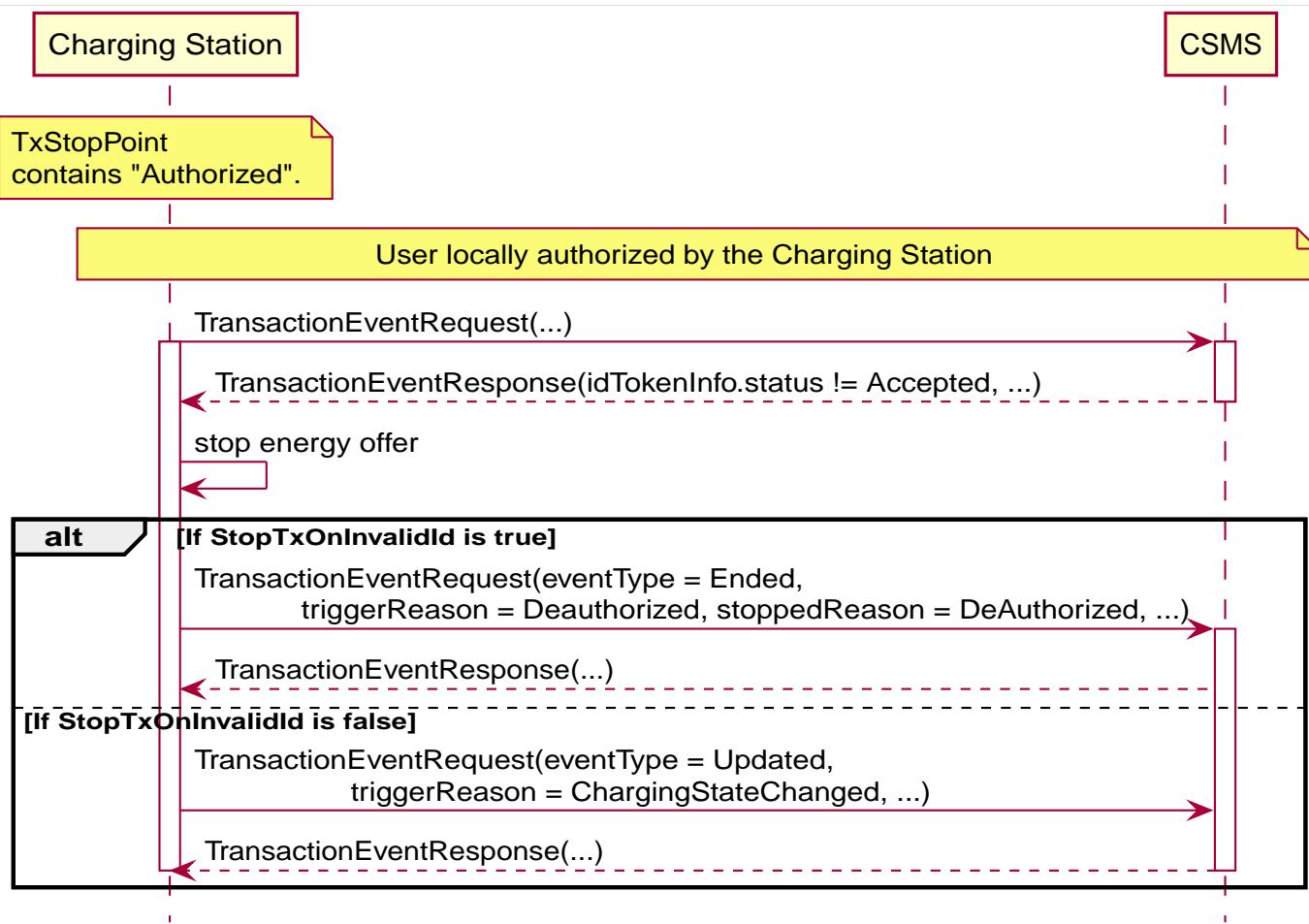


Figure 52. Sequence Diagram: Stop Transaction options - Deauthorized

S5	Scenario objective	Stop a transaction when the EV driver is no longer authorized and/or the EV is disconnected.
	Scenario description	1. The Charging Station is disconnected from EV and/or the EV driver is no longer authorized. 2. The Charging Station sends a TransactionEventRequest (<code>eventType = Ended</code>) notifying the CSMS about a transaction that has ended. 3. The CSMS responds with a TransactionEventResponse , confirming that the TransactionEventRequest was received.
	Prerequisite(s)	A transaction is ongoing. Configuration Variable: <code>TxStopPoint</code> contains: <code>PowerPathClosed</code>
	Postcondition(s)	Successful postcondition: The transaction is ended and the CSMS is <i>Successfully informed</i> . Failure postcondition: The transaction is still ongoing. or The CSMS is <i>not informed</i> .

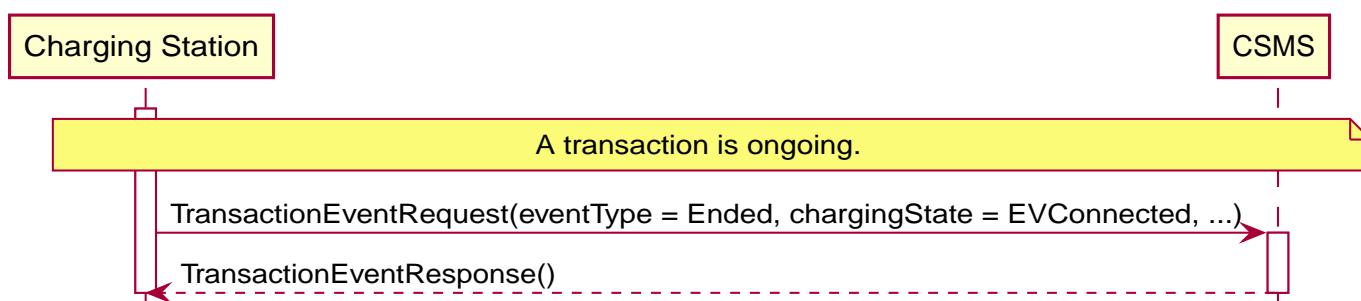


Figure 53. Sequence Diagram: Stop Transaction options - PowerPathClosed

S6	Scenario objective	Stop a transaction when energy transfer stops. This will also mean the transaction stops when the EV stops taking energy, for example when the battery is to hot.
	Scenario description	<p>1. The energy transfer between EV and the Charging Station stops (for example: EV stops charging).</p> <p>2. The Charging Station sends a TransactionEventRequest (<code>eventType = Ended</code>) notifying the CSMS about a transaction that has ended.</p> <p>3. The CSMS responds with a TransactionEventResponse, confirming that the TransactionEventRequest was received.</p>
	Prerequisite(s)	<p>A transaction is ongoing.</p> <p>Configuration Variable: TxStopPoint contains: EnergyTransfer</p>
	Postcondition(s)	<p>Successful postcondition: The transaction is ended and the CSMS is <i>Successfully informed</i>.</p> <p>Failure postcondition: The transaction is still ongoing. or The CSMS is <i>not informed</i>.</p>

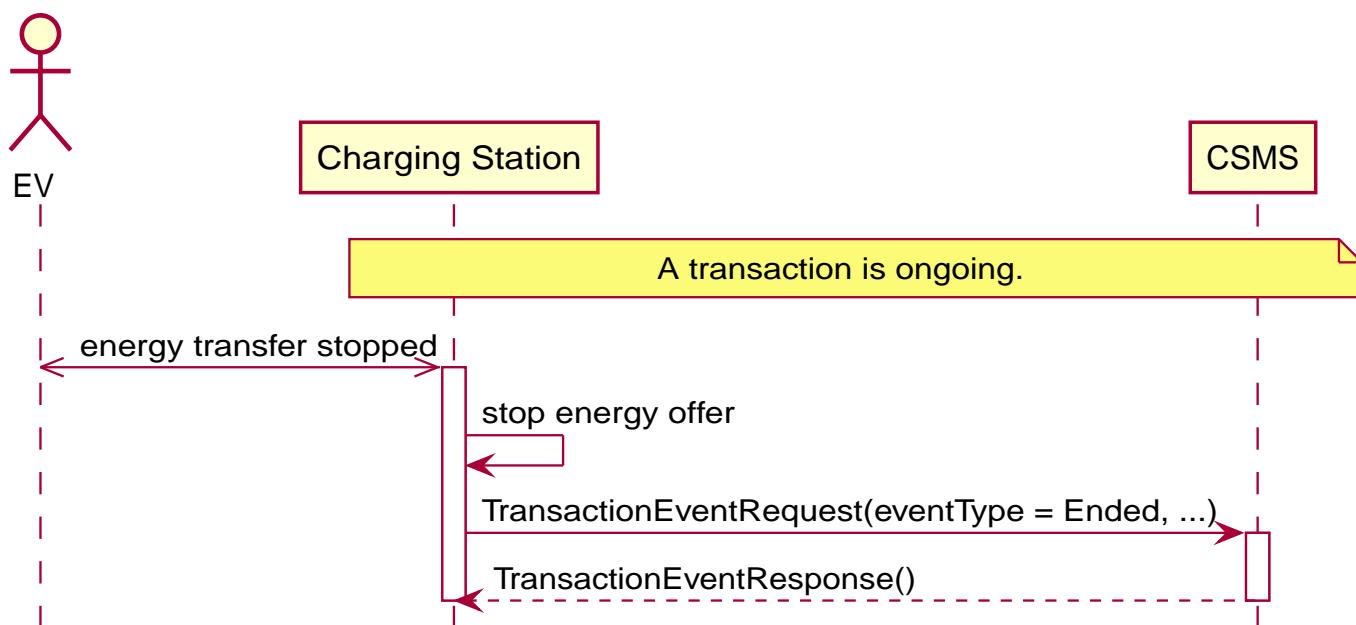


Figure 54. Sequence Diagram: Stop Transaction options - EnergyTransfer

S7	Scenario objective	Stop a transaction when EV driver ends authorization
	Scenario description	<p>1. The EV drivers presents an IdToken to end the charging.</p> <p>2. The Charging Station sends a TransactionEventRequest (<code>eventType = Ended</code>) notifying the CSMS about a transaction that has ended.</p> <p>3. The CSMS responds with a TransactionEventResponse, confirming that the TransactionEventRequest was received.</p>
	Prerequisite(s)	<p>A transaction is ongoing.</p> <p>Configuration Variable: TxStopPoint contains: Authorized (or PowerPathClosed).</p>
	Postcondition(s)	<p>Successful postcondition: The transaction is ended and the CSMS is <i>Successfully informed</i>.</p> <p>Failure postcondition: The transaction is still ongoing. or The CSMS is <i>not informed</i>.</p>

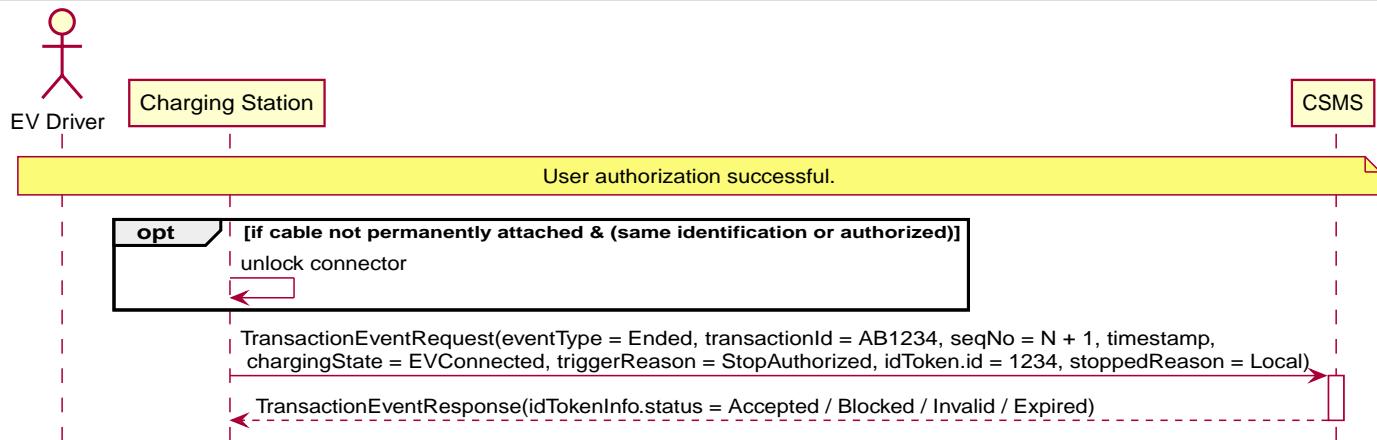


Figure 55. Sequence Diagram: Stop Transaction options - Authorized

7	Error handling	n/a
8	Remark(s)	n/a

E06 - Stop Transaction options - Requirements

Table 107. E06 - Requirements

ID	Precondition	Requirement definition
E06.FR.01	TxStopPoint contains: ParkingBayOccupancy AND Parking Bay Detector no longer detects the "EV"	The Charging Station SHALL stop the transaction and send a TransactionEventRequest (eventType = Ended) to the CSMS.
E06.FR.02	TxStopPoint contains: EVConnected AND Connection between Charging Station and EV is lost.	The Charging Station SHALL stop the transaction and send a TransactionEventRequest (eventType = Ended) to the CSMS.
E06.FR.03	TxStopPoint contains: Authorized AND EV Driver is authorized to stop a transaction.	The Charging Station SHALL stop the transaction and send a TransactionEventRequest (eventType = Ended) to the CSMS.
E06.FR.04	TxStopPoint contains: Authorized AND CSMS returns a non-valid idTokenInfo in a TransactionEventResponse	The Charging Station SHALL stop the transaction and send a TransactionEventRequest (eventType = Ended) to the CSMS.
E06.FR.05	TxStopPoint contains: DataSigned AND Charging Station can no longer retrieve signed meter values.	The Charging Station SHALL stop the transaction and send a TransactionEventRequest (eventType = Ended) to the CSMS.
E06.FR.06	TxStopPoint contains: PowerPathClosed AND (Connection between Charging Station and EV is lost OR Authorization has ended or idToken is deauthorized)	The Charging Station SHALL stop the transaction and send a TransactionEventRequest (eventType = Ended) to the CSMS.
E06.FR.07	TxStopPoint contains: EnergyTransfer AND Energy transfer stops	The Charging Station SHALL stop the transaction and send a TransactionEventRequest (eventType = Ended) to the CSMS.
E06.FR.08	If a transaction is not ended by the EV Driver at the Charging Station	The Charging Station SHALL include the stoppedReason element in the TransactionEventRequest(eventType = Ended). What reason to use is described in the description of reasonEnumType.

ID	Precondition	Requirement definition
E06.FR.09	If a transaction is ended by the EV Driver at the Charging Station (e.g. EV Driver presented IdToken to stop the transaction)	The Charging Station MAY omit the stoppedReason element in the TransactionEventRequest (eventType = Ended) (hence the CSMS can interpret the reason as local when omitted).
E06.FR.10	As part of the normal transaction termination.	The Charging Station SHALL unlock the cable (if not permanently attached).
E06.FR.11	When configured to send meter data in the TransactionEventRequest (eventType = Ended), See: Meter Values - Configuration	The Charging Station SHALL add the configured measurands to the optional meterValue field with context = Transaction.End in the TransactionEventRequest (eventType = Ended) sent to the CSMS to provide more details about transaction usage.
E06.FR.12	E06.FR.11 AND The Charging Station is running low on memory	The Charging Station MAY drop meter data in the TransactionEventRequest (eventType = Ended) message.
E06.FR.13	E06.FR.12	When dropping meter data, the Charging Station SHALL drop intermediate values first (1st value, 3th value, 5th etc), not start dropping values from the start of the list or stop adding values to the list.
E06.FR.14	When a TransactionEventRequest has to be created	The Charging Station SHALL set the message's seqNo field as specified in Sequence Number Generation .
E06.FR.15	When sending a TransactionEventRequest	The Charging Station SHALL set the triggerReason to inform the CSMS about what triggered the event. What reason to use is described in the description of TriggerReasonEnumType .
E06.FR.16	A transaction was stopped by an Abnormal Error or Fault Condition.	The Charging Station SHALL send TransactionEventRequest (eventType = Ended , triggerReason = AbnormalCondition)_ to the CSMS.

E07 - Transaction locally stopped by IdToken

Table 108. E07 - Transaction locally stopped by IdToken

No.	Type	Description
1	Name	Transaction locally stopped by IdToken
2	ID	E07
	<i>Functional block</i>	E. Transactions
3	Objective(s)	The EV Driver wants to stop an ongoing transaction, by locally presenting his IdToken.
4	Description	This use case covers how the EV Driver can stop a transaction when he wants to leave the charging station.
	Actors	Charging Station, CSMS, EV Driver
	<i>Scenario description</i> TxStopPoint = Authorized (or PowerPathClosed)	<p><i>Transaction ends with triggerReason=StopAuthorized when ending authorization:</i></p> <ol style="list-style-type: none"> 1. The EV Driver presents IdToken a second time to end charging. 2. The Charging Station sends a TransactionEventRequest (eventType = Ended) with triggerReason = StopAuthorized and stoppedReason = Local. 3. The CSMS responds with a TransactionEventResponse. 4. The Charging Station stops the energy transfer and if the cable is not permanently attached, the Charging Station unlocks the cable.
	<i>Alternative scenario(s)</i> TxStopPoint = Authorized (or PowerPathClosed)	<p><i>Transaction ends with triggerReason=ChargingStateChanged when stopping charging:</i></p> <ol style="list-style-type: none"> 1. The EV Driver presents IdToken a second time to end charging. 2. The Charging Station sends a TransactionEventRequest (eventType = Updated) with triggerReason = StopAuthorized 3. The CSMS responds with a TransactionEventResponse. 4. The Charging Station stops the energy transfer and if the cable is not permanently attached, the Charging Station unlocks the cable. 5. The Charging Station sends a TransactionEventRequest (eventType = Ended) with triggerReason = ChargingStateChanged, transactionInfo.chargingState = EVConnected 6. The CSMS responds with a TransactionEventResponse.
5	Prerequisite(s)	A transaction is ongoing.

No.	Type	Description
6	Postcondition(s)	<p>Successful postcondition: The CSMS has received all relevant information about the transaction and the Charging Station is in <i>Idle</i> status.</p> <p>Failure postcondition: The transaction is still ongoing or the Charging Station is in Idle status and still holds information about the transaction that it has to deliver to the CSMS.</p>

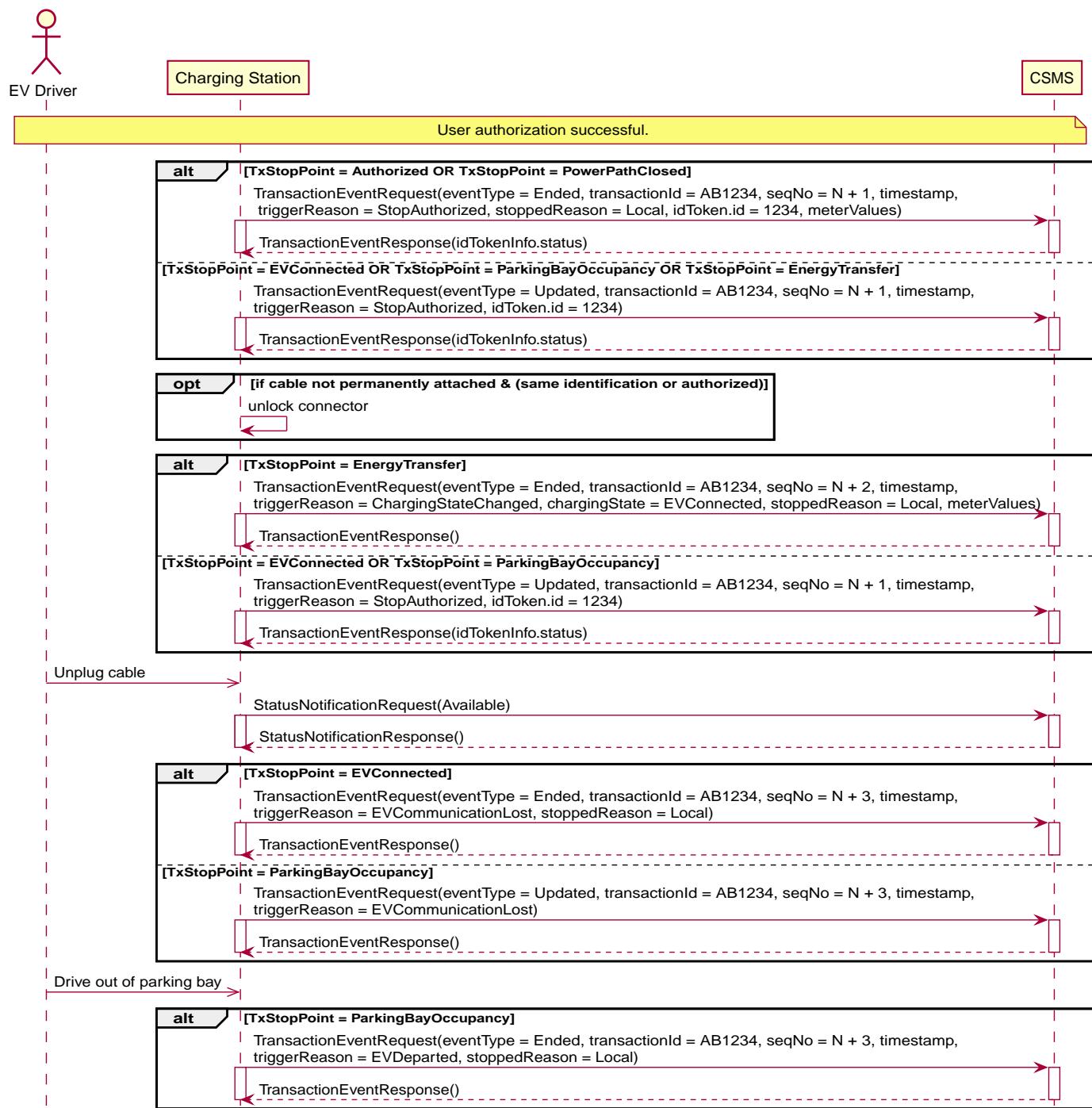


Figure 56. Sequence Diagram: *Transaction locally stopped by IdToken with TransactionEventRequest reported strictly by TxStopPoint configuration*

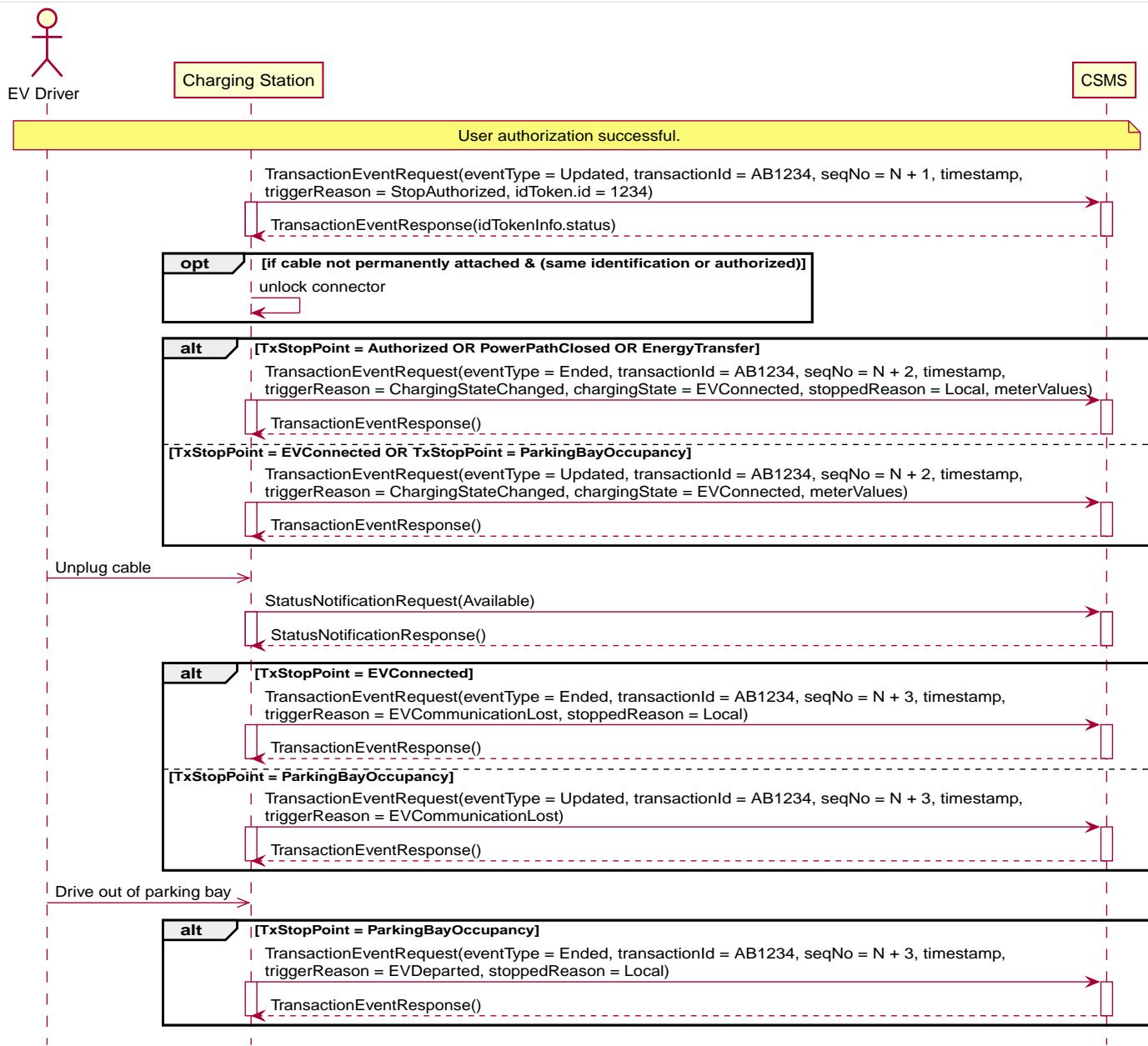


Figure 57. Sequence Diagram: Transaction locally stopped by IdToken with delayed TransactionEventRequest eventType = Ended for TxStopPoint = Authorized OR PowerPathClosed

7	Error handling	n/a
8	Remark(s)	<p>The scenario descriptions are based on TxStopPoint containing Authorized or PowerPathClosed. The sequence diagrams also show behavior for other TxStopPoint values in the alt-blocks.</p> <p>The CSMS cannot prevent a transaction from stopping.</p>

E07 - Transaction locally stopped by IdToken - Requirements

Table 109. E07 - Requirements

ID	Precondition	Requirement definition	Note
E07.FR.01	When an idToken is presented during a transaction that has been authorized AND (a) the presented idToken is the same as the idToken that started the authorization OR (b) when the presented idToken is in the Local Authorization List or Authorization Cache AND is valid AND has the same GroupIdToken as the IdToken that started the authorization.	The Charging Station SHALL end the authorization of the transaction, without first sending an AuthorizeRequest	The idToken that started the authorization can always be used to end the authorization. Ending authorization will end delivery of energy. Depending on the TxStopPoint ending of the authorization may also end the transaction. (See C01.FR.03)
E07.FR.02	E07.FR.01	The Charging Station SHALL send a TransactionEventRequest with <i>triggerReason</i> = StopAuthorized and SHOULD include the idToken used to stop authorization.	The stopping <i>idToken</i> may differ from the starting <i>idToken</i> , when they share the same GroupId.
E07.FR.04	If a transaction is ended in a normal way.	The stoppedReason element MAY be omitted.	e.g. EV-driver presented IdToken to stop the transaction.
E07.FR.05	If a transaction is ended in a normal way	The stoppedReason SHOULD be assumed 'Local'.	e.g. EV-driver presented IdToken to stop the transaction.
E07.FR.06	If the transaction is <i>not</i> ended normally.	stoppedReason SHOULD be set to a correct value.	
E07.FR.07	As part of the normal transaction termination.	The Charging Station SHALL unlock the cable (if not permanently attached).	
E07.FR.08	When configured to send meter data in the TransactionEventRequest (<i>eventType</i> = Ended), See: Meter Values - Configuration	The Charging Station SHALL add the configured measurands to the optional <i>meterValue</i> field with <i>context</i> = Transaction.End in the TransactionEventRequest (<i>eventType</i> = Ended) sent to the CSMS to provide more details about transaction usage.	
E07.FR.09	E07.FR.08 AND The Charging Station is running low on memory	The Charging Station MAY drop meter data in the TransactionEventRequest (<i>eventType</i> = Ended) message.	
E07.FR.10	E07.FR.09	When dropping meter data, the Charging Station SHALL drop intermediate values first (1st value, 3th value, 5th etc), not start dropping values from the start of the list or stop adding values to the list.	
E07.FR.11	When a TransactionEventRequest has to be created	The Charging Station SHALL set the message's seqNo field as specified in Sequence Number Generation .	This enables the CSMS to track the completeness of transaction information
E07.FR.12	SampledDataSignReadings is true	The Charging Station SHALL retrieve signed meter values and put them in the <i>signedMeterValue</i> field of sampledValues .	

E08 - Transaction stopped while Charging Station is offline

Table 110. E08 - Transaction stopped while Charging Station is offline

No.	Type	Description
1	Name	Transaction stopped while Charging Station is offline
2	ID	E08
	<i>Functional block</i>	E. Transactions
	<i>Parent use case</i>	E07 - Local Stop Transaction
3	Objective(s)	To enable the EV Driver to stop a transaction while the Charging Station is <i>Offline</i> .
4	Description	This use case describes how an EV Driver can stop a transaction while the Charging Station is <i>Offline</i> . While a transaction is ongoing and the Charging Station is <i>Offline</i> , the EV Driver presents his IdToken, if the Charging Stations knows locally (without asking the CSMS) that this IdToken is allowed to stop the transaction, it will stop the ongoing transaction. When the Charging Station restores the connection with the CSMS, it needs to send the information about this <i>Offline</i> stop transaction to the CSMS.
	Actors	Charging Station, CSMS, EV Driver
	<i>Scenario description</i>	<ol style="list-style-type: none"> 1. The EV Driver presents IdToken to stop the transaction. 2. When this is the same IdToken as was used to start the transaction, or via the Local Authorization List and / or Authorization Cache the GroupId can be validated: the transaction is stopped. 3. The Charging Station stops the energy offer. 4. The TransactionEventRequest (eventType = Ended) is stored/queued by the Charging Station. 5. The connection between Charging Station and CSMS is restored. 6. The Charging Station starts to send queued messages 7. The stored TransactionEventRequest is sent, notifying the CSMS about the transaction that was stopped.
5	Prerequisite(s)	Transaction ongoing and connection lost.
6	Postcondition(s)	Charging Station is in <i>Idle</i> status.

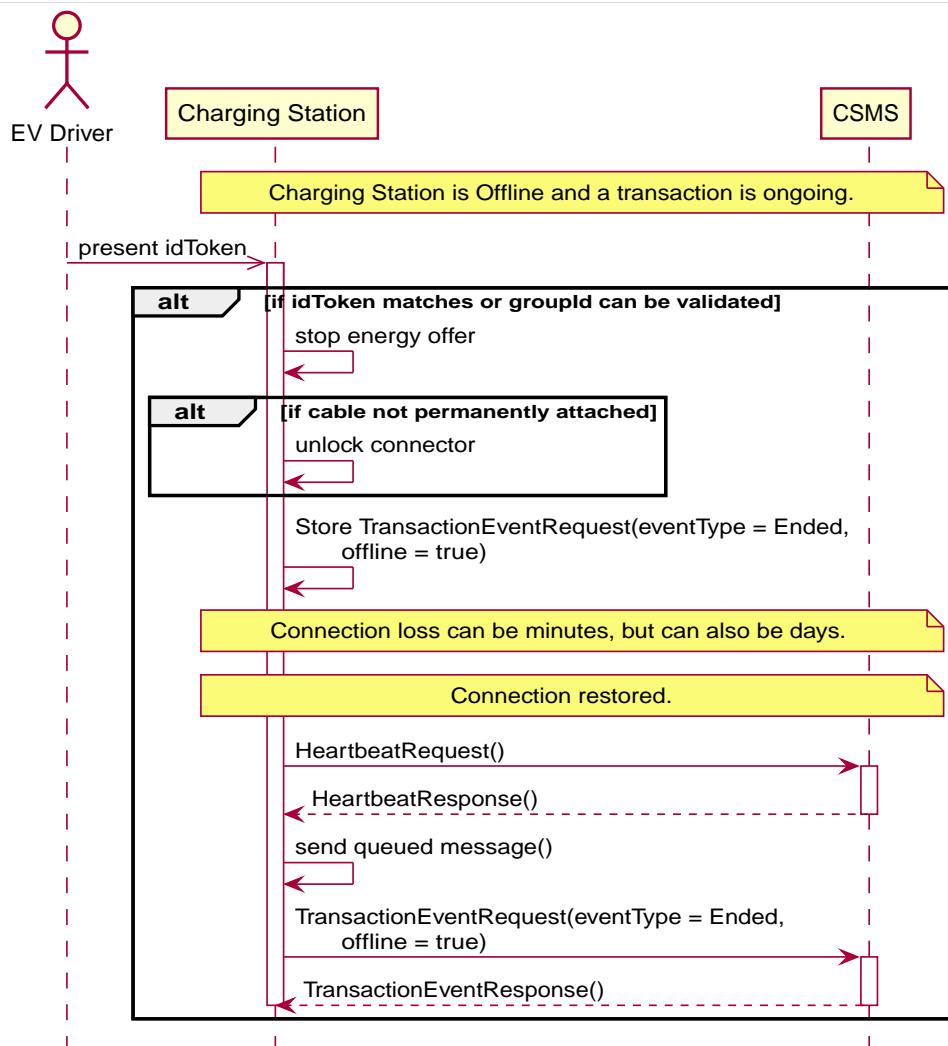


Figure 58. Sequence Diagram: Transaction stopped while Charging Station is offline

7	Error handling	n/a
8	Remark(s)	<p>groupId check must be done on Local Authorization List and / or Authorization Cache if available.</p> <p>The scenario description and sequence diagram above are based on the Configuration Variable for stop transaction being configured as follows.</p> <p>TxStopPoint: ParkingBayOccupancy, EVConnected, Authorized</p> <p>This use-case is also valid for other configurations, but then the transaction might stop at another moment, which might change the sequence in which message are sent. For more details see the use case: E06 - Stop Transaction options</p>

E08 - Transaction stopped while Charging Station is offline - Requirements

Table 111. E08 - Requirements

ID	Precondition	Requirement definition	Note
E08.FR.01	If the IdToken presented is the same as the IdToken used to start the transaction.	The Charging Station SHALL stop the energy offering.	
E08.FR.02	If the IdToken presented has the same GroupId as the IdToken used to start the transaction.	The Charging Station SHALL stop the energy offering.	
E08.FR.03	(E08.FR.01 OR E08.FR.02) AND Cable not permanently attached	The Charging Station SHALL unlock the connector.	
E08.FR.04	(E08.FR.01 OR E08.FR.02)	The Charging Station SHALL "generate" a TransactionEventRequest (eventType = Ended).	

ID	Precondition	Requirement definition	Note
E08.FR.05	When Offline.	The Charging Station MUST queue any TransactionEventRequest messages.	
E08.FR.06	After the connection is restored.	The Charging Station MUST send queued TransactionEventRequest messages.	
E08.FR.07		The flag: <i>offline</i> SHALL be set to TRUE for any TransactionEventRequest that occurred while the Charging Station was offline.	
E08.FR.08	When a TransactionEventRequest has to be created	The Charging Station SHALL set the message's seqNo field as specified in Sequence Number Generation .	This enables the CSMS to track the completeness of transaction information.
E08.FR.09	When configured to send meter data in the TransactionEventRequest (<i>eventType</i> = Ended), See: Meter Values - Configuration	The Charging Station SHALL add the configured measurands to the optional <i>meterValue</i> field in the TransactionEventRequest (<i>eventType</i> = Ended) sent to the CSMS to provide more details about transaction usage.	
E08.FR.10	E08.FR.09 AND The Charging Station is running low on memory	The Charging Station MAY drop meter data in the TransactionEventRequest (<i>eventType</i> = Ended) message.	
E08.FR.11	E08.FR.10	When dropping meter data, the Charging Station SHALL drop intermediate values first (1st value, 3th value, 5th etc), not start dropping values from the start of the list or stop adding values to the list.	
E08.FR.12	SampledDataSignReadings is true	The Charging Station SHALL retrieve signed meter values and put them in the <i>signedMeterValue</i> field of sampledValues .	

E09 - When cable disconnected on EV-side: Stop Transaction

Table 112. E09 - When cable disconnected on EV-side: Stop Transaction

No.	Type	Description
1	Name	When cable disconnected on EV-side: Stop Transaction
2	ID	E09
	<i>Functional block</i>	E. Transactions
	<i>Parent use case</i>	E07 - Local Stop Transaction
3	Objective(s)	To stop an ongoing transaction when the Charging Cable is unplugged on the EV side.
4	Description	<p>This use case covers how a transaction is stopped when the EV Driver unplugs the cable at the EV side. In this use case the Configuration Variable: <code>StopTxOnEVSideDisconnect</code> = true.</p> <p>The Charging Cable is unplugged at the EV side. This is detected by the Charging Station. The Charging Station stops the transaction and sends a <code>TransactionEventRequest</code> to the CSMS. The Charging Cable, if locked and <code>UnlockOnEvSideDisconnect</code> = false, will remain locked at the Charging Station until the EV Driver returns and presents his/hers IdToken. Otherwise it will unlock the cable.</p>
	Actors	Charging Station, CSMS, EV Driver
	<i>Scenario description</i>	<ol style="list-style-type: none"> 1. The cable is unplugged at the EV. 2. The energy offer is suspended. 3. The Charging Station sends <code>TransactionEventRequest</code> (<code>eventType</code> = Ended, <code>stoppedReason</code> = <code>EVDisconnected</code>) to the CSMS. 4. The CSMS responds with <code>TransactionEventResponse</code>. 5. The EV Driver is authorized and unplugs the cable. 6. The Charging Station sends <code>StatusNotificationRequest</code> to the CSMS with the status <code>Available</code>. 7. The CSMS responds with <code>StatusNotificationResponse</code>.
	<i>Alternative scenario(s)</i>	E09 - When cable disconnected on EV-side: Suspend Transaction
5	Prerequisite(s)	<p>Configuration Variable: <code>StopTxOnEVSideDisconnect</code> = true</p> <p>A transaction is ongoing</p>
6	Postcondition(s)	<p>Successful postcondition: The Charging Station is in <code>Idle</code> status.</p> <p>Failure postcondition: n/a</p>

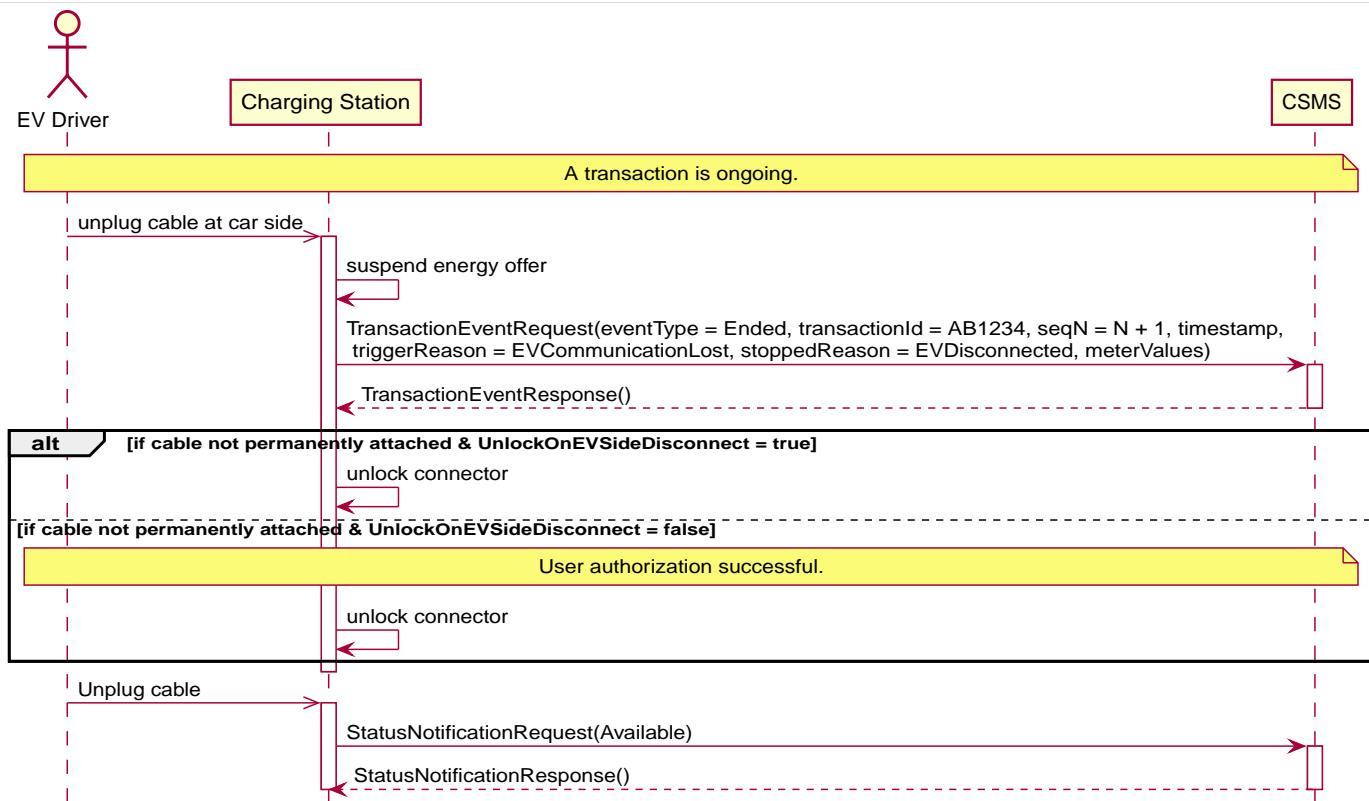


Figure 59. Sequence Diagram: When cable disconnected on EV-side: Stop Transaction

7	Error handling	n/a
8	Remark(s)	<p>When the Charging Cable is plugged back in, the charging will not resume/continue.</p> <p>The scenario description and sequence diagram above are based on the Configuration Variable for stop transaction being configured as follows.</p> <p><code>TxStopPoint: Authorized</code></p> <p>This use-case is also valid for other configurations, but then the transaction might stop at another moment, which might change the sequence in which messages are sent. For more details see the use case: E06 - Stop Transaction options</p>

E09 - When cable disconnected on EV-side: Stop Transaction - Requirements

Table 113. E09 - Requirements

ID	Precondition	Requirement definition	Note
E09.FR.01	If <code>StopTxOnEVSideDisconnect</code> = true .	The transaction SHALL be deauthorized when the cable is disconnected from the EV. If the EV is reconnected, energy transfer is not allowed until the transaction is authorized once again.	Setting <code>StopTxOnEVSideDisconnect</code> to true will prevent sabotage acts when unplugging not locked cables on EV side.
E09.FR.02	E09.FR.01 AND the cable is not permanently attached AND <code>UnlockOnEvSideDisconnect</code> = true.	The Charging Station SHALL unlock the Charging Cable.	

ID	Precondition	Requirement definition	Note
E09.FR.03	E09.FR.01 AND the cable is not permanently attached AND <code>UnlockOnEvSideDisconnect = false.</code>	The Charging Station SHALL unlock the Charging Cable only after authorization by the EV Driver.	
E09.FR.04	When a TransactionEventRequest has to be created	The Charging Station SHALL set the message's seqNo field as specified in Sequence Number Generation .	This enables the CSMS to track the completeness of transaction information
E09.FR.05	When configured to send meter data in the TransactionEventRequest (eventType = Ended) , See: Meter Values - Configuration	The Charging Station SHALL add the configured measurands to the optional meterValue field in the TransactionEventRequest(eventType = Ended) sent to the CSMS to provide more details about transaction usage.	
E09.FR.06	E09.FR.05 AND The Charging Station is running low on memory	The Charging Station MAY drop meter data in the TransactionEventRequest(eventType = Ended) message.	
E09.FR.07	E09.FR.06	When dropping meter data, the Charging Station SHALL drop intermediate values first (1st value, 3th value, 5th etc), not start dropping values from the start of the list or stop adding values to the list.	
E09.FR.08	<code>SampledDataSignReadings is true</code>	The Charging Station SHALL retrieve signed meter values and put them in the <i>signedMeterValue</i> field of sampledValues.	

E10 - When cable disconnected on EV-side: Suspend Transaction

Table 114. E10 - When cable disconnected on EV-side: Suspend Transaction

No.	Type	Description
1	Name	When cable disconnected on EV-side: Suspend Transaction
2	ID	E10
	Functional block	E. Transactions
	Parent use case	E07 - Local Stop Transaction
3	Objective(s)	To suspend an ongoing transaction when the Charging Cable is unplugged on the EV side.
4	Description	<p>This use case covers how a transaction is suspended when the EV Driver unplugs the cable at the EV side. In this use case the Configuration Variable: <code>StopTxOnEVSideDisconnect</code> = false.</p> <p>The Charging Cable is unplugged at the EV side. This is detected by the Charging Station. The Charging Station stops the energy offering (safety), but does not stop the transaction. The Charging Cable, if locked, will remain locked at the Charging Station until the EV Driver returns and presents his/hers IdToken.</p>
	Actors	Charging Station, CSMS, EV Driver
	Scenario description	<p>1. EV Driver unplugs the cable at the EV while a transaction is ongoing. 2. The energy offer is suspended.</p> <p><i>If the EV Driver plugs the cable back in, the transaction is resumed.</i></p> <p>A1. The Charging Station sends a TransactionEventRequest (<code>eventType</code> = <code>Updated</code>, <code>trigger</code> = <code>CablePluggedIn</code>)</p> <p>A2. The CSMS responds with a TransactionEventResponse.</p> <p><i>If cable not permanently attached</i></p> <p>B1. The EV Driver is authorized by the Charging Station and/or CSMS to unlock the charging cable.</p> <p>B2. The Cable is unlocked.</p> <p>B3. The Charging Station sends a TransactionEventRequest (<code>eventType</code> = <code>Ended</code>, <code>trigger</code> = <code>StopAuthorized</code>).</p> <p>B4. The EV Driver removes the charging cable.</p> <p>B5. The Charging Station sends a StatusNotificationRequest to the CSMS with the status <code>Available</code>.</p> <p>B6. The CSMS responds with a StatusNotificationResponse.</p> <p><i>If cable permanently attached</i></p> <p>C1. The Cable is not plugged in within timeout.</p> <p>C2. The Charging Station sends a TransactionEventRequest (<code>eventType</code> = <code>Ended</code>, <code>trigger</code> = <code>EVCommunicationLost</code>, <code>stoppedReason</code> = <code>EVDisconnected</code>).</p> <p>C3. The Charging Station sends a StatusNotificationRequest to the CSMS with the status <code>Available</code>.</p> <p>C4. The CSMS responds with a StatusNotificationResponse.</p>
	Alternative scenario(s)	E09 - When cable disconnected on EV-side: Stop Transaction
5	Prerequisite(s)	Configuration Variable: <code>StopTxOnEVSideDisconnect</code> = false A transaction is ongoing
6	Postcondition(s)	<p>Successful postcondition: The Charging Station is in <code>Idle</code> status. The regular transaction is resumed.</p> <p>Failure postcondition: n/a</p>

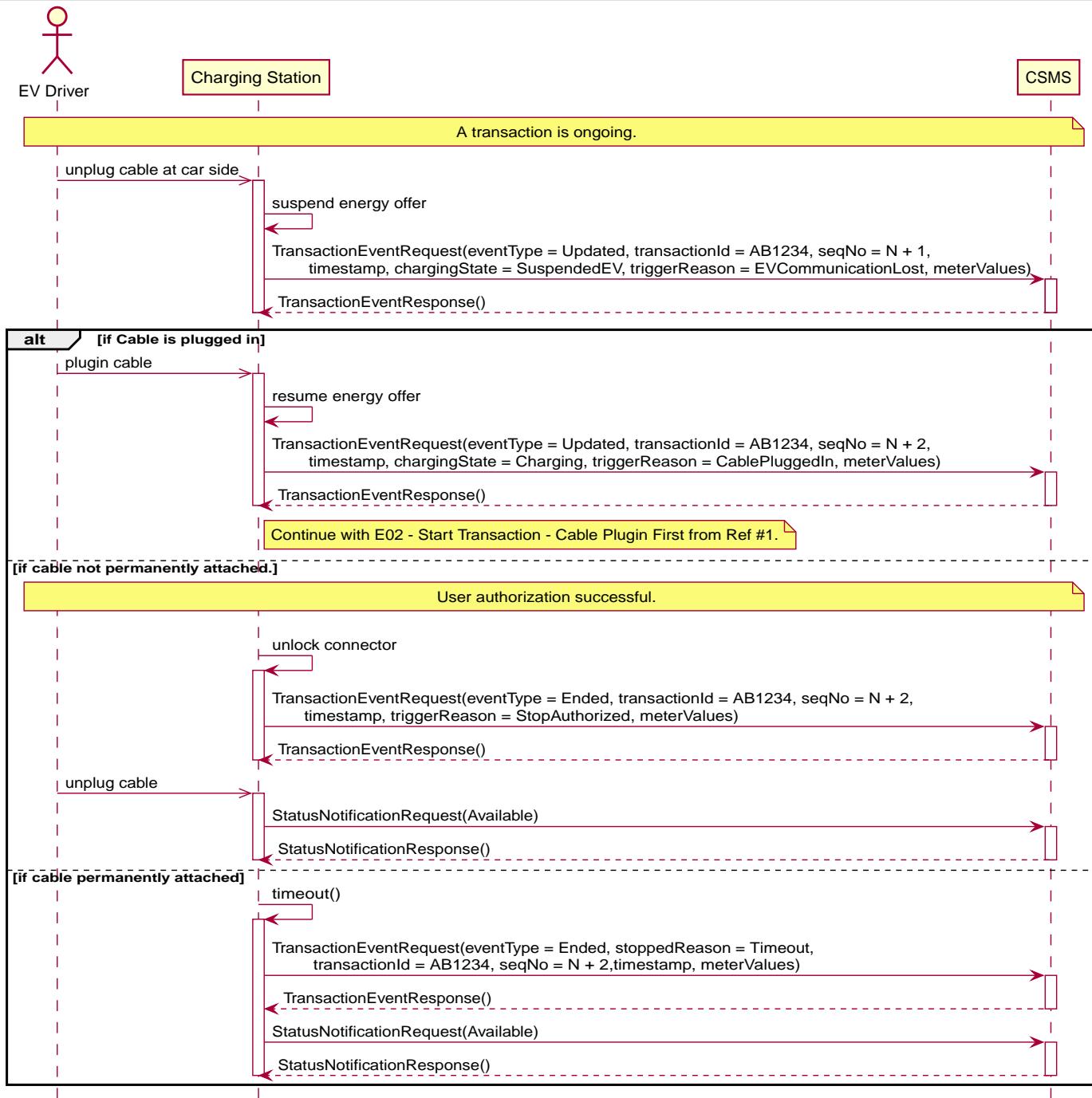


Figure 60. Sequence Diagram: When cable disconnected on EV-side: Suspend Transaction

7	Error handling	n/a
8	Remark(s)	<p>When the Charging Cable is plugged back in, the charging is resumed.</p> <p>When the cable is permanently attached and the cable is not plugged in within a certain timeout, the Charging Station stops the transaction. This timeout is not defined by OCPP, it is left to the implementor of the Charging Station.</p> <p>The scenario description and sequence diagram above are based on the Configuration Variable for stop transaction being configured as follows.</p> <p>TxStopPoint: ParkingBayOccupancy, Authorized</p> <p>This use-case is also valid for other configurations, but then the transaction might stop at another moment, which might change the sequence in which message are sent. For more details see the use case: E06 - Stop Transaction options</p>

E10 - When cable disconnected on EV-side: Suspend Transaction - Requirements

Table 115. E10 - Requirements

ID	Precondition	Requirement definition	Note
E10.FR.01	Cable not permanently attached	The Connector SHALL remain locked at the Charging Station until the EV Driver presents the IdToken.	
E10.FR.02	Cable permanently attached AND Cable not plugged in within timeout	The Charging Station SHALL deauthorize the transaction.	
E10.FR.03	When a TransactionEventRequest has to be created	The Charging Station SHALL set the message's seqNo field as specified in Sequence Number Generation .	This enables the CSMS to track the completeness of transaction information
E10.FR.04	When configured to send meter data in the TransactionEventRequest (eventType = Ended) , See: Meter Values - Configuration	The Charging Station SHALL add the configured measurands to the optional meterValue field in the TransactionEventRequest(eventType = Ended) sent to the CSMS to provide more details about transaction usage.	
E10.FR.05	E10.FR.04 AND The Charging Station is running low on memory	The Charging Station MAY drop meter data in the TransactionEventRequest(eventType = Ended) message.	
E10.FR.06	E10.FR.05	When dropping meter data, the Charging Station SHALL drop intermediate values first (1st value, 3th value, 5th etc), not start dropping values from the start of the list or stop adding values to the list.	
E10.FR.07	SampledDataSignReadings is true	The Charging Station SHALL retrieve signed meter values and put them in the <i>signedMeterValue</i> field of sampledValues.	

E11 - Connection Loss During Transaction

Table 116. E11 - Connection Loss During Transaction

No.	Type	Description
1	Name	Connection Loss During Transaction
2	ID	E11
	Functional block	E. Transactions
3	Objective(s)	To enable a Charging Station to continue a transaction while the Charging Station loses its connection
4	Description	This use cases describes how a Charging Station can continue an ongoing transaction while losing and regaining the connection with the CSMS.
	Actors	Charging Station, CSMS
	Scenario description	<ol style="list-style-type: none"> 1. The connection of the Charging Station is lost, while a transaction is ongoing. 2. The transaction events of the Charging Station are stored. 3. The connection with the CSMS is restored. 4. The Charging Station sends the stored transaction events to the CSMS using TransactionEventRequest (offline = TRUE). 5. The Charging Station resumes regular communication.
	Alternative scenario(s)	E04 - Offline Start Transaction
5	Prerequisite(s)	Transaction ongoing and connection lost.
6	Postcondition(s)	<p>Successful postcondition: The Charging Station resumes regular communication.</p> <p>Failure postcondition: n/a</p>

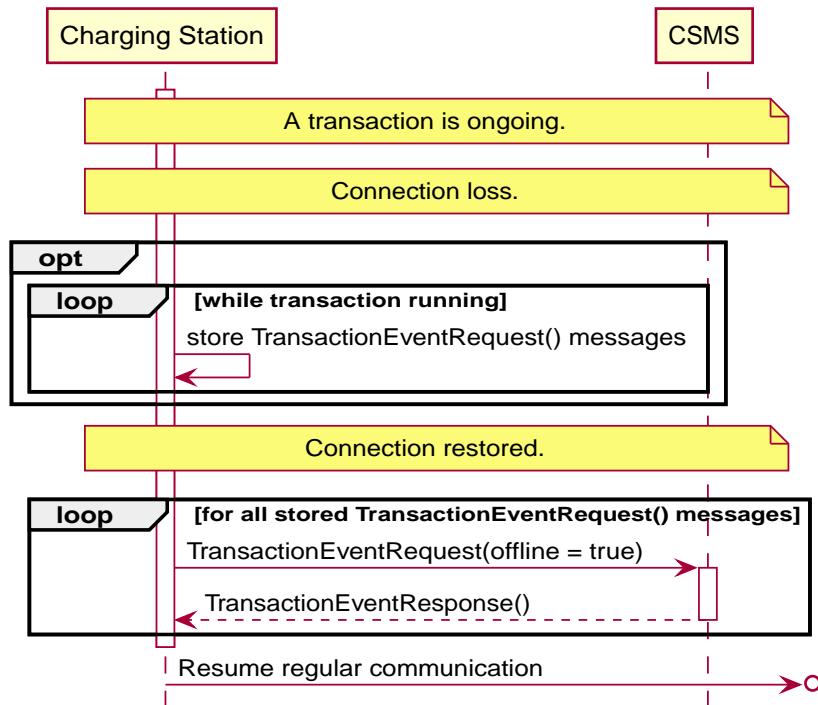


Figure 61. Sequence Diagram: Connection Loss During Transaction

7	Error handling	n/a
8	Remark(s)	n/a

E11 - Connection Loss During Transaction - Requirements

Table 117. E11 - Requirements

ID	Precondition	Requirement definition
E11.FR.01	When Offline	The Charging Station MUST queue all TransactionEventRequest messages, that it would have sent to the CSMS if the Charging Station had been online.
E11.FR.02	After the connection is restored.	The Charging Station MUST send queued TransactionEventRequest messages with the flag <i>offline</i> set to TRUE.
E11.FR.03	When configured to send meter data in the TransactionEventRequest(eventType = Updated) , See: Meter Values - Configuration	The Charging Station SHALL add the configured measurands to the optional <i>meterValue</i> field in the TransactionEventRequest(eventType = Updated) sent to the CSMS to provide more details during the transaction.
E11.FR.04	E11.FR.03 AND <i>Offline</i> AND The Charging Station is running low on memory	The Charging Station MAY drop TransactionEventRequest(eventType = Updated) messages.
E11.FR.05	E11.FR.04	When dropping TransactionEventRequest(eventType = Updated) messages, the Charging Station SHALL drop intermediate messages first (1st message, 3th message, 5th message etc.), not start dropping messages from the start or stop adding messages to the queue.
E11.FR.06	E11.FR.03 AND Amount of meter data is too much for 1 TransactionEventRequest(eventType = Updated)	The Charging Station MAY split the meter data over multiple TransactionEventRequest(eventType = Updated) messages with the same <i>timestamp</i> .
E11.FR.07		If the Charging Station goes offline, every message that is still in the queue SHALL be set <i>Offline</i> .
E11.FR.08	SampledDataSignReadings is true	The Charging Station SHALL retrieve signed meter values and put them in the <i>signedMeterValue</i> field of sampledValues .

E12 - Inform CSMS of an Offline Occurred Transaction

Table 118. E12 - Inform CSMS of an Offline Occurred Transaction

No.	Type	Description
1	Name	Inform CSMS of an Offline Occurred Transaction
2	ID	E12
	Functional block	E. Transactions
3	Objective(s)	To enable the Charging Station to inform the CSMS that a transaction occurred while the Charging Station was <i>Offline</i> .
4	Description	This use case covers how the Charging Station starts and stops a transaction since connection loss.
	Actors	Charging Station, CSMS
	Scenario description	<ol style="list-style-type: none"> 1. The connection with the CSMS is restored. 2. The Charging Station sends a Heartbeat message to the CSMS. 3. The Charging Station sends TransactionEventRequest (<code>eventType = Started</code>, <code>offline = TRUE</code>) to the CSMS. 4. The CSMS responds with TransactionEventResponse, accepting the transaction. 5. The Charging Station sends TransactionEventRequest (<code>eventType = Updated</code>, <code>offline = TRUE</code>) 6. The CSMS responds with TransactionEventResponse. 7. The Charging Station sends TransactionEventRequest (<code>eventType = Ended</code>, <code>offline = TRUE</code>) 8. The CSMS responds with TransactionEventResponse.
5	Prerequisite(s)	At least one <i>Offline</i> transaction has taken place.
6	Postcondition(s)	<p>Successful postcondition: The CSMS has processed all transactions that occurred <i>Offline</i>.</p> <p>Failure postcondition: n/a</p>

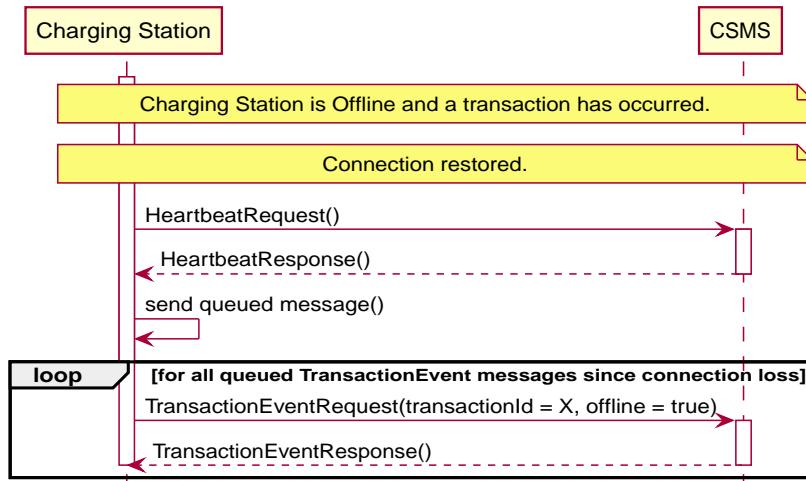


Figure 62. Sequence Diagram: Inform CSMS of an Offline Occurred Transaction

7	Error handling	n/a
8	Remark(s)	n/a

E12 - Inform CSMS of an Offline Occurred Transaction - Requirements

Table 119. E12 - Requirements

ID	Precondition	Requirement definition
E12.FR.01	When Offline	The Charging Station MUST queue all TransactionEventRequest messages, that it would have sent to the CSMS if the Charging Station had been online.

ID	Precondition	Requirement definition
E12.FR.02	After the connection is restored.	The Charging Station MUST send queued TransactionEventRequest messages with the flag <i>offline</i> set to TRUE.
E12.FR.03	When configured to send meter data in the TransactionEventRequest(eventType = Updated) , See: Meter Values - Configuration	The Charging Station SHALL add the configured measurands to the optional <i>meterValue</i> field in the TransactionEventRequest(eventType = Updated) sent to the CSMS to provide more details during the transaction.
E12.FR.04	E12.FR.03 AND <i>Offline</i> AND The Charging Station is running low on memory	The Charging Station MAY drop TransactionEventRequest(eventType = Updated) messages.
E12.FR.05	E12.FR.04	When dropping TransactionEventRequest(eventType = Updated) messages, the Charging Station SHALL drop intermediate messages first (1st message, 3th message, 5th message etc.), not start dropping messages from the start or stop adding messages to the queue.
E12.FR.06	E12.FR.03 AND Amount of meter data is too much for 1 TransactionEventRequest(eventType = Updated)	The Charging Station MAY split the meter data over multiple TransactionEventRequest(eventType = Updated) messages with the same <i>timestamp</i> .
E12.FR.07	When configured to send meter data in the TransactionEventRequest(eventType = Ended) , See: Meter Values - Configuration	The Charging Station SHALL add the configured measurands to the optional <i>meterValue</i> field in the TransactionEventRequest(eventType = Ended) sent to the CSMS to provide more details about transaction usage.
E12.FR.08	E12.FR.07 AND The Charging Station is running low on memory	The Charging Station MAY drop meter data in the TransactionEventRequest(eventType = Ended) message.
E12.FR.09	E12.FR.08	When dropping meter data, the Charging Station SHALL drop intermediate values first (1st value, 3th value, 5th etc), not start dropping values from the start of the list or stop adding values to the list.
E12.FR.10	SampledDataSignReadings is true	The Charging Station SHALL retrieve signed meter values and put them in the <i>signedMeterValue</i> field of sampledValues .

E13 - Transaction-related message not accepted by CSMS

Table 120. E13 - Transaction-related message not accepted by CSMS

No.	Type	Description
1	Name	Transaction-related message not accepted by CSMS
2	ID	E13
	Functional block	E. Transactions
3	Objective(s)	To define how a Charging Station shall handle not accepted messages.
4	Description	There are situations/issues why a CSMS might not accept a transaction related message, or does not reply within the MessageTimeout. Most are error scenarios. When something like this happens, the Charging Station SHALL retry the messages a couple of times.
	Actors	Charging Station, CSMS
	Scenario description	<ol style="list-style-type: none"> 1. The Charging Station sends a transaction-related message to the CSMS. 2. The message is not accepted and <code>MessageAttemptsTransactionEvent</code> not reached. 3. The Charging Station waits the number of preceding transmissions of this same message times <code>MessageAttemptIntervalTransactionEvent</code> seconds. 4. The Charging Station resends the transaction-related message to the CSMS.
5	Prerequisite(s)	n/a
6	Postcondition(s)	<p>Successful postcondition: MessageAttemptsTransactionEvent is not reached AND the transaction-related message is accepted. MessageAttemptsTransactionEvent is reached AND the transaction-related message is disposed.</p> <p>Failure postcondition: MessageAttemptsTransactionEvent is not reached AND the transaction-related message is disposed. MessageAttemptsTransactionEvent is reached AND the transaction-related message is accepted.</p>

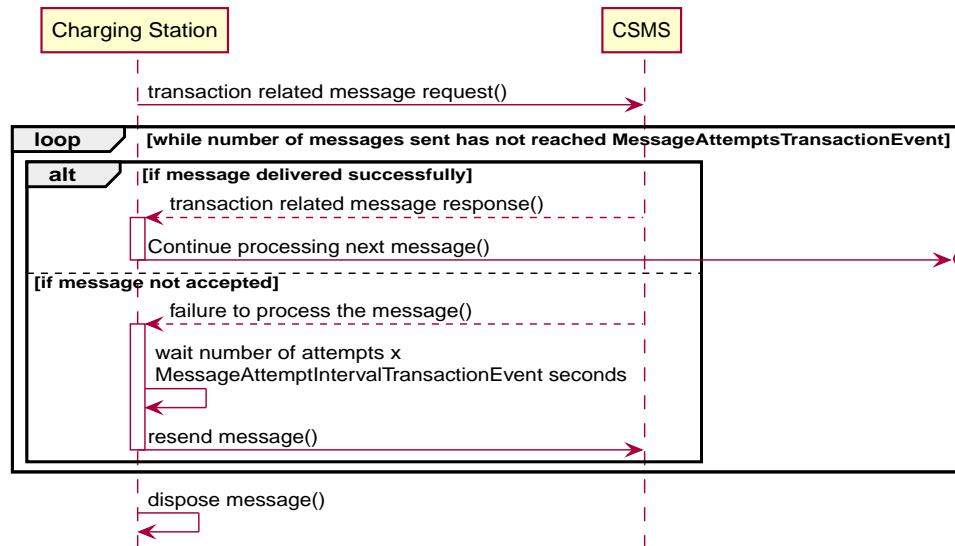


Figure 63. Sequence Diagram: Transaction-related message not accepted by CSMS

7	Error handling	n/a
8	Remark(s)	This use case describes the expected behaviour when the CSMS does not accept a message, or does not reply within the message timeout, this is different from a situation where the communication between Charging Station and CSMS is Offline.

E13 - Transaction-related message not accepted by CSMS - Requirements

Table 121. E13 - Requirements

ID	Precondition	Requirement definition
E13.FR.01		The number of times and the interval with which the Charging Station should retry such failed transaction-related messages MAY be configured using the MessageAttemptsTransactionEvent and MessageAttemptIntervalTransactionEvent Configuration Variables.
E13.FR.02	When the Charging Station encounters a first failure to deliver a certain transaction-related message.	The Charging Station SHALL send this message again as long as it keeps resulting in a failure to process this message and it has not yet encountered as many failures to process this message for this message as specified in its MessageAttemptsTransactionEvent Configuration Variable.
E13.FR.03	The CSMS does not accept a transaction-related message.	The Charging Station SHALL wait as many seconds as specified in its MessageAttemptIntervalTransactionEvent key, multiplied by the number of preceding transmissions of this same message.
E13.FR.04	If the final attempt fails.	The Charging Station SHALL discard the message and continue with the next transaction-related message, if there is any.

E13 - Transaction-related message not accepted by CSMS - Example

As an example, consider a Charging Station that has the value "3" for the [MessageAttemptsTransactionEvent](#) Configuration Variable and the value "60" for the [MessageAttemptIntervalTransactionEvent](#) Configuration Variable. It sends a [TransactionEventRequest](#) message and detects a failure to process the message in the CSMS. The Charging Station SHALL wait for 60 seconds, and resend the message. In the case when there is a second failure, the Charging Station SHALL wait for 120 seconds, before resending the message. If this final attempt fails, the Charging Station SHALL discard the message and continue with the next transaction-related message, if there is any.

E14 - Check transaction status

No.	Type	Description
1	Name	Check transaction status
2	ID	E14
	Functional block	E. Transactions
3	Objectives	To enable the CSMS to request the status of a transaction and to find out whether there are queued transaction-related messages.
4	Description	There are scenarios where a CSMS needs to know whether there are still messages for a transaction that need to be delivered. For example: A CSMS receives a TransactionEventRequest (<code>eventType = Ended</code>), it wants to start the billing process for this transaction but detects it is still missing some intermediate messages (it can check this via the sequence number in the messages). It can ask if the Charging Station has still messages in the queue for this transaction with the GetTransactionStatusRequest specifying the transactionId. Depending on the result the CSMS might for example: wait for the messages to be delivered, or start the billing process without the information. It may also need to know whether a transaction is still ongoing. If the CSMS wants to know if there are transaction-related messages in the queue at all (not just for a specific transaction), it can send a GetTransactionStatusRequest without a transactionId.
	Actors	CSMS, Charging Station
	Scenario description	<ol style="list-style-type: none"> 1. The CSMS sends a GetTransactionStatusRequest with or without a transactionId to the Charging Station. 2. The Charging Station responds with a GetTransactionStatusResponse.
5	Prerequisites	The CSMS knows the transactionId of a transaction it wants to know the status of.
6	Postcondition(s)	<p>Successful postcondition: The CSMS knows the status of the requested transaction.</p> <p>Failure postcondition: The CSMS does not know the status of the requested transaction.</p>

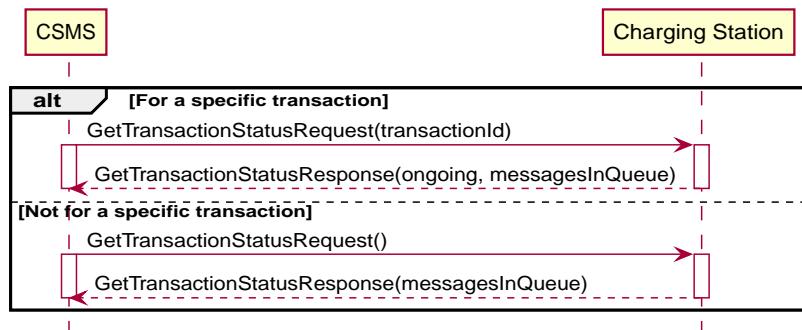


Figure 64. Sequence Diagram: Check transaction status

7	Error Handling	n/a
8	Remarks	When the CSMS receives a GetTransactionStatusResponse with both fields (<code>ongoingIndicator</code> and <code>messagesInQueue</code>) set to false, this might mean that the transaction is finished and there are no more messages in the queue for this transaction, or the Charging Station doesn't know anything about this transaction (anymore).

E14 - Check transaction status - Requirements

ID	Precondition	Requirements
E14.FR.01	The Charging Station receives a GetTransactionStatusRequest with a transactionId AND It did not do a transaction with that transactionId	The Charging Station SHALL respond with <code>ongoingIndicator = false</code> AND <code>messagesInQueue = false</code> .
E14.FR.02	The Charging Station receives a GetTransactionStatusRequest with a transactionId AND The transaction with that transactionId has not stopped yet	The Charging Station's response SHALL have <code>ongoingIndicator = true</code> .

ID	Precondition	Requirements
E14.FR.03	The Charging Station receives a GetTransactionStatusRequest with a transactionId AND The transaction with that transactionId has stopped	The Charging Station's response SHALL have <i>ongoingIndicator</i> = false.
E14.FR.04	The Charging Station receives a GetTransactionStatusRequest with a transactionId AND It has transaction-related messages to be delivered about the transaction with that transactionId	The Charging Station's response SHALL have <i>messagesInQueue</i> = true.
E14.FR.05	The Charging Station receives a GetTransactionStatusRequest with a transactionId AND It has no transaction-related messages to be delivered about the transaction with that transactionId	The Charging Station's response SHALL have <i>messagesInQueue</i> = false.
E14.FR.06	The Charging Station receives a GetTransactionStatusRequest without a transactionId	The Charging Station's response SHALL NOT have <i>ongoingIndicator</i> set.
E14.FR.07	The Charging Station receives a GetTransactionStatusRequest without a transactionId AND It has transaction-related messages to be delivered	The Charging Station's response SHALL have <i>messagesInQueue</i> = true.
E14.FR.08	The Charging Station receives a GetTransactionStatusRequest without a transactionId AND It has no transaction-related messages to be delivered	The Charging Station's response SHALL have <i>messagesInQueue</i> = false.

2.2. Interrupting and Stopping ISO 15118 Charging

E15 - End of charging process

Table 122. E15 - End of charging process

No.	Type	Description
1	Name	End of charging process.
2	ID	E15
	<i>Functional block</i>	E. Transactions
	<i>Reference</i>	ISO15118-1 H1 - End of charging process
3	Objectives	See ISO15118-1 , use case Objective H1, page 44.
4	Description	See ISO15118-1 , use case Description H1, page 44.
5	Actors	EV, EVSE, EV Driver
6	Scenario Description	See ISO15118-1 , use case Description H1, Basic elementary use case description, first 5 bullets and last 2 remarks, page 44. 6. The EV driver unplugs the cable from the EV 7. The Charging Station sends a TransactionEventRequest with eventType <i>eventType</i> = Ended to the CSMS.
7	Prerequisites	See ISO15118-1 , use case Prerequisites H1, page 44.
8	Postcondition(s)	The CSMS has received all relevant information about the transaction. See ISO15118-1 , use case End Conditions H1, page 44.

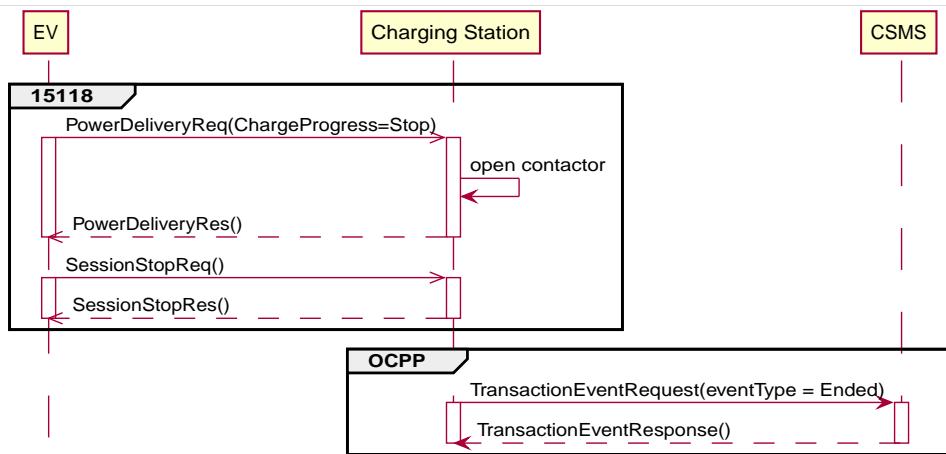


Figure 65. End of charging process

9	Error handling	n/a
10	Remark(s)	<p>See ISO15118-1, use case Requirements H1, page 44 for the trigger.</p> <p>The scenario description and sequence diagram above are based on the Configuration Variable for stop transaction being configured as follows.</p> <p>TxStopPoint: ParkingBayOccupancy, EVConnected, Authorized, DataSigned, PowerPathClosed</p> <p>This use-case is also valid for other configurations, but then the transaction might stop at another moment, which might change the sequence in which message are sent. For more details see the use case: E06 - Stop Transaction options</p>

Source: [ISO15118-1](#)

E15 - End of charging process - Requirements

Table 123. E15 - Requirements

ID	Precondition	Requirement definition
E15.FR.01	When configured to send meter data in the TransactionEventRequest (eventType = Ended) , See: Meter Values - Configuration	The Charging Station SHALL add the configured measurands to the optional meterValue field in the TransactionEventRequest (eventType = Ended) sent to the CSMS to provide more details about transaction usage.
E15.FR.02	E15.FR.01 AND The Charging Station is running low on memory	The Charging Station MAY drop meter data in the TransactionEventRequest(eventType = Ended) message.
E15.FR.03	E15.FR.02	When dropping meter data, the Charging Station SHALL drop intermediate values first (1st value, 3th value, 5th etc), not start dropping values from the start of the list or stop adding values to the list.
E15.FR.04		After receiving a SessionStopReq message from the EV, the CS SHALL send a TransactionEventRequest message with eventType = Ended to inform the CSMS that the charging transaction has been stopped (by the EV).

F. RemoteControl

1. Introduction

This Functional Block describes three types of use cases for remote control management from the CSMS:

1. Remote Transaction Control. These use cases describe the functionality which enable the CSO (or indirect a third party) to start/stop a transaction with a remote command.
2. Unlocking a Connector. These use cases describe the functionality that enables the CSO (or indirect a third party) to unlock the Connector with a remote command. This can for example be used to assist customers when they have problems unplugging their cable.
3. Remote Trigger. These use cases describe all the remote trigger functionality of OCPP. This functionality enables remote triggering of messages. For example, requesting messages to be resend or request current status of some ongoing processes in the Charging Station.

2. Use cases & Requirements

2.1. Remote Transaction Control

F01 - Remote Start Transaction - Cable Plugin First

Table 124. F01 - Remote Start Transaction - Cable Plugin First

No.	Type	Description
1	Name	Remote Start Transaction - Cable Plugin First
2	ID	F01
	Functional block	F. Remote Control
3	Objective(s)	<ul style="list-style-type: none"> 1. To remotely start a transaction by the CSMS. 2. To enable a CSO to help an EV Driver that has problems starting a transaction. 3. To enable third parties (e.g. mobile apps) to control charging transactions via the CSMS.
4	Description	This use case describes how the CSMS remotely requests the Charging Station to start a transaction by sending RequestStartTransactionRequest . Upon receipt, the Charging Station responds with RequestStartTransactionResponse and a status indicating whether it is able to try to start a transaction or not.
	Actors	Charging Station, CSMS, CSO
	Scenario description	<ul style="list-style-type: none"> 1. The EV Driver plugs in the cable at the Charging Station. 2. The Charging Station sends a StatusNotificationRequest to the CSMS to inform it about a Connector that became <i>Occupied</i>. 3. The CSMS responds with a StatusNotificationResponse, confirming that the StatusNotificationRequest was received. 4. The Charging Station sends a TransactionEventRequest (<code>eventType = Started</code>) notifying the CSMS about a transaction that has started (even when the driver is not yet known.) 5. The CSMS responds with a TransactionEventResponse, confirming that the TransactionEventRequest was received. 6. An external trigger (as example in this use case: EV Driver) triggers the remote start. 7. The CSMS sends a RequestStartTransactionRequest to the Charging Station. 8. The Charging Station responds with a RequestStartTransactionResponse with the <code>transactionId</code> of the already started transaction to the CSMS. 9. Optionally: the EV Driver is authorized by the CSMS. 10. The Charging Station sends a TransactionEventRequest (<code>eventType = Updated, chargingState = Charging</code>) message to inform the CSMS that the charging has started.
	Alternative scenario(s)	Remote Start Transaction - Remote Start First F02 - Remote Start Transaction - Remote Start First
5	Prerequisite(s)	Charging Cable plugged in first.
6	Postcondition(s)	The Charging Station offers energy. If the value of AuthorizeRemoteStart is <code>true</code> , the Charging Station will only offer energy when it successfully authorized the IdToken, using Local Authorization List , Authorization Cache and/or an AuthorizeRequest .

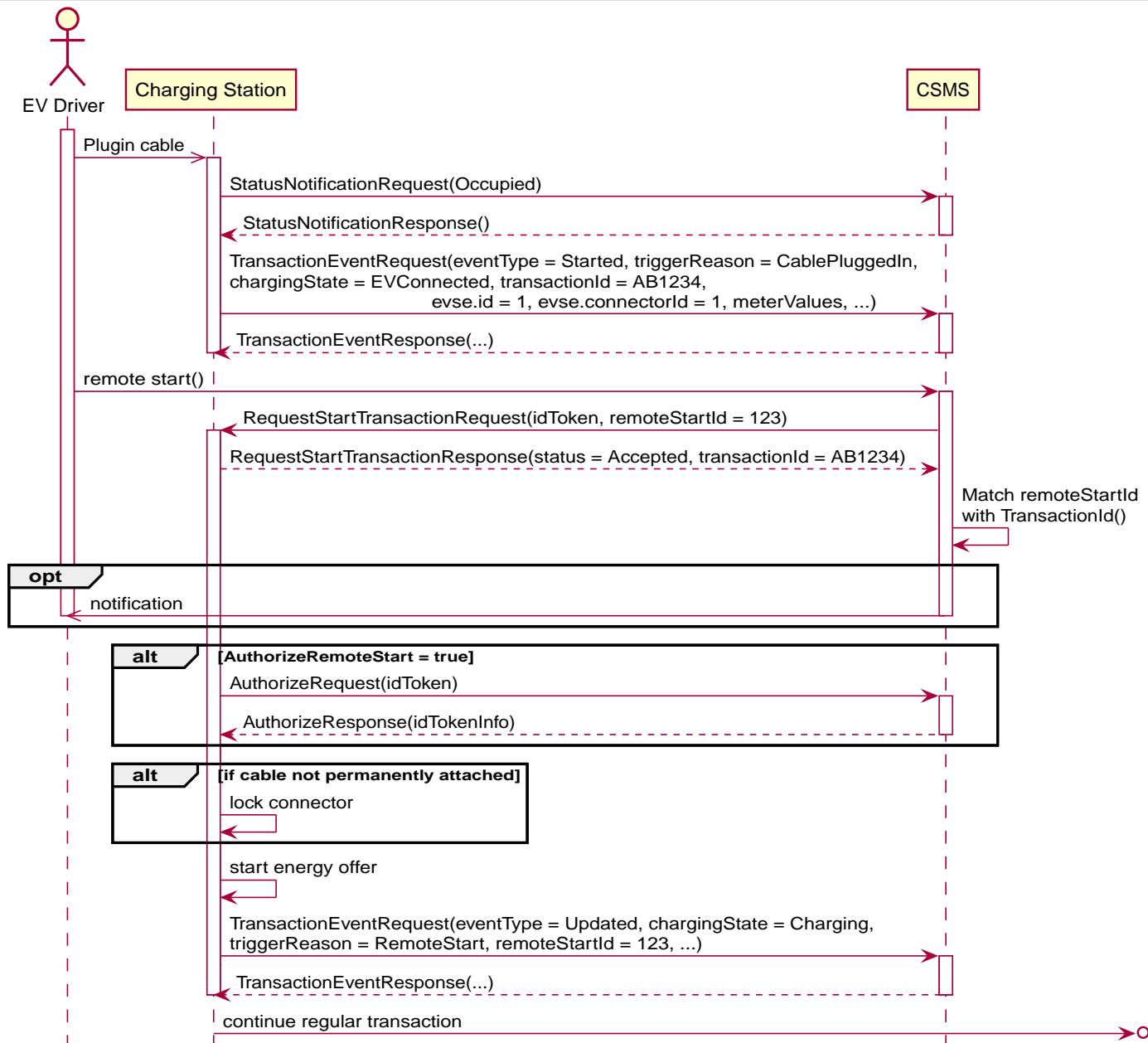


Figure 66. Sequence Diagram: Remote Start Transaction - Cable Plugged in First

7	Error handling	n/a
8	Remark(s)	<p>An external trigger can be e.g. a Charging Station Operator or an EV Driver app.</p> <p>The <code>RequestStartTransactionResponse</code> contains a status which indicates whether the Charging Station has accepted the request and will attempt to start a transaction.</p> <p>The CSMS is allowed to send a <code>RequestStartTransactionRequest</code> with <code>IdTokenType</code> of type: <code>NoAuthorization</code>. The operator should be aware that if the Charging Station supports local stop transaction, this transaction can be stopped by anyone.</p> <p>The scenario description and sequence diagram above are based on the Configuration Variable for start transaction being configured as follows: <code>TxStartPoint: EVConnected, Authorized, DataSigned, PowerPathClosed, EnergyTransfer</code> This use-case is also valid for other configurations, but then the transaction might start/stop at another moment, which might change the sequence in which message are sent. For more details see the use cases: E01 - Start Transaction options.</p>

F01 - Remote Start Transaction - Cable Plugin First - Requirements

Table 125. F01 - Requirements

ID	Precondition	Requirement definition	Note
F01.FR.01	If the value of AuthorizeRemoteStart = true AND Charging Station receives a RequestStartTransactionRequest	The Charging Station SHALL behave as if in response to a local action at the Charging Station to allow energy transfer after successful authorization of the IdToken given in RequestStartTransactionRequest message.	Charging Station will first respond to the request and then try to authorize the IdToken, using the Local Authorization List, Authorization Cache and/or an AuthorizeRequest . Energy transfer is only allowed after authorization was obtained.
F01.FR.02	If the value of AuthorizeRemoteStart = false AND Charging Station receives a RequestStartTransactionRequest	The Charging Station SHALL allow energy transfer for the IdToken given in RequestStartTransactionRequest message without checking authorization.	Charging Station will first respond to the request, and send a TransactionEventRequest with the idToken to the CSMS, and the CSMS will check the authorization status of the IdToken when processing this TransactionEventRequest .
F01.FR.03	F01.FR.01 OR F01.FR.02	The Charging Station SHALL send a TransactionEventRequest to the CSMS, and the CSMS will check the authorization status of the IdToken when processing this TransactionEventRequest .	If CSMS returns an authorization status that is not Accepted, then Charging Station must stop energy transfer as per use case E05.
F01.FR.04		RequestStartTransactionRequest SHALL contain an IdToken, which Charging Station SHALL use, if it is able to start a transaction, in the TransactionEventRequest sent to the CSMS.	
F01.FR.05		The transaction SHALL be started in the same way as described in E01 - Start Transaction - Cable Plugin First .	
F01.FR.06		RequestStartTransactionRequest MAY contain an evseld if the transaction is to be started on a specific EVSE.	When no evseld is provided, the Charging Station is in control of the EVSE selection.
F01.FR.07	If the RequestStartTransactionRequest does not contain an evseld.	The Charging Station MAY reject the RequestStartTransactionRequest .	
F01.FR.08		The CSMS MAY include a ChargingProfile in the RequestStartTransactionRequest .	
F01.FR.09	F01.FR.08	The purpose of this ChargingProfile SHALL be set to TxProfile .	
F01.FR.10	F01.FR.08	The Charging Station SHALL use this ChargingProfile for the transaction that is started by this RequestStartTransaction.	
F01.FR.11	F01.FR.08	The transactionId in the ChargingProfile SHALL NOT be set.	
F01.FR.12	If a Charging Station without support for Smart Charging receives a RequestStartTransactionRequest with a ChargingProfile .	The Charging Station SHALL ignore the specified ChargingProfile .	The device model variable SmartChargingCtrlr.Enabled tells CSMS whether smart charging is supported.

ID	Precondition	Requirement definition	Note
F01.FR.13	When a transaction is created on the Charging Station, but has not been authorized. AND RequestStartTransactionRequest is received.	The Charging Station SHALL return the <i>transactionId</i> in the RequestStartTransactionResponse .	
F01.FR.14	When configured to send meter data in the TransactionEventRequest (<i>eventType</i> = Started), See: Meter Values - Configuration	The Charging Station SHALL add the configured measurands to the optional <i>meterValue</i> field in the TransactionEventRequest (<i>eventType</i> = Started) sent to the CSMS to provide more details during the transaction.	
F01.FR.15	When configured to send meter data in the TransactionEventRequest (<i>eventType</i> = Updated), See: Meter Values - Configuration	The Charging Station SHALL add the configured measurands to the optional <i>meterValue</i> field in the TransactionEventRequest (<i>eventType</i> = Updated) sent to the CSMS to provide more details during the transaction.	
F01.FR.16	F01.FR.15 AND Amount of meter data is too much for 1 TransactionEventRequest (<i>eventType</i> = Updated)	The Charging Station MAY split meter data over multiple TransactionEventRequest (<i>eventType</i> = Updated) messages with the same <i>timestamp</i> .	
F01.FR.17	When sending a TransactionEventRequest	The Charging Station SHALL set the <i>triggerReason</i> to inform the CSMS about what triggered the event. What reason to use is described in the description of TriggerReasonEnumType .	
F01.FR.18	After a transaction has been started	The Charging Station MAY send additional TransactionEventRequest (<i>eventType</i> = Updated) messages during the transaction when a <i>trigger event</i> occurs.	
F01.FR.19	When a RequestStartTransactionRequest is received.	The next TransactionEventRequest SHALL contain <i>triggerReason</i> : RemoteStart .	
F01.FR.20	If the RequestStartTransactionRequest does not contain an <i>evselId</i> AND the Charging Station is capable of selecting an EVSE	The Charging Station SHALL select an EVSE to be used as a value for <i>evselId</i> for the operation	See also F01.FR.07 if Charging Station does not support starting at an arbitrary EVSE.
F01.FR.21	When the <i>evselId</i> for RequestStartTransactionRequest is Reserved for an <i>idToken</i> that differs from <i>idToken</i> in the request AND has no reservation for a <i>groupIdToken</i>	The Charging Station SHALL respond with RequestStartTransactionResponse with <i>status</i> = Rejected.	
F01.FR.22	When the <i>evselId</i> for RequestStartTransactionRequest is Reserved for an <i>idToken</i> that differs from <i>idToken</i> in the request AND is Reserved for a <i>groupIdToken</i> that differs from <i>groupIdToken</i> in the request	The Charging Station SHALL respond with RequestStartTransactionResponse with <i>status</i> = Rejected.	EV is not allowed to use station if neither <i>idToken</i> nor <i>idGroupToken</i> match the reservation.
F01.FR.23	When the evse for RequestStartTransactionRequest is Unavailable or Faulted	The Charging Station SHALL respond with RequestStartTransactionResponse with <i>status</i> = Rejected.	
F01.FR.24	When the <i>evselId</i> for RequestStartTransactionRequest is Occupied AND this <i>evselId</i> has a transaction that has been authorized	The Charging Station SHALL respond with RequestStartTransactionResponse with <i>status</i> = Rejected.	Only an EVSE with no transaction or with a transaction that has not yet been authorized can be matched with the RequestStartTransaction Request
F01.FR.25	F01.FR.13	The Charging Station SHALL put the <i>remoteStartId</i> in the next TransactionEventRequest it sends for the associated transaction.	

ID	Precondition	Requirement definition	Note
F01.FR.26	If a Charging Station with support for Smart Charging receives a RequestStartTransactionRequest with an invalid ChargingProfile .	The Charging Station SHALL respond with RequestStartTransactionResponse with status = Rejected and optionally with <i>reasonCode</i> = "InvalidProfile" or "InvalidSchedule".	The device model variable SmartChargingCtrlr.Enabled tells CSMS whether smart charging is supported.

F02 - Remote Start Transaction - Remote Start First

Table 126. F02 - Remote Start Transaction - Remote Start First

No.	Type	Description
1	Name	Remote Start Transaction - Remote Start first
2	ID	F02
	<i>Functional block</i>	F. Remote Control
	<i>Parent use case</i>	F01 - Remote Start Transaction - Cable Plugin First
3	Objective(s)	To enable the CSMS to remotely start a transaction while the RequestStartTransactionRequest is sent first, before the connection between Charging Station and EV is established.
4	Description	This use case covers how the CSMS is able to remotely start a transaction for the User.
	Actors	Charging Station, CSMS, External Trigger
	<i>Scenario description</i>	<p>1. An External Trigger triggers the remote start.</p> <p>2. The CSMS sends RequestStartTransactionRequest to the Charging Station.</p> <p>3. The Charging Station responds with RequestStartTransactionResponse to the CSMS.</p> <p>4. The EV Driver is authorized by the CSMS, dependent on the Configuration Variable settings.</p> <p>5. The Charging Station sends a TransactionEventRequest (<i>eventType</i> = Started) notifying the CSMS about a transaction that has started</p> <p>6. The cable is plugged in.</p> <p>6a. Charging Station sends a StatusNotificationRequest with <i>Occupied</i>.</p> <p>6b. CSMS sends a StatusNotificationResponse to the Charging Station</p> <p>7. The energy offer is started.</p> <p>8. The Charging Station sends a TransactionEventRequest (<i>eventType</i> = Updated, <i>chargingState</i> = Charging) message to inform the CSMS that the charging has started.</p> <p>9. The CSMS sends TransactionEventResponse to the Charging Station</p>
5	Prerequisite(s)	Charging Cable not plugged in. Remote start first. Enable mobile apps to control charging transactions via the CSMS.
6	Postcondition(s)	<p>Successful postcondition: The transaction for which a start was request has started and the EV is charging.</p> <p>Failure postcondition: The transaction for which a start was request did not start or the EV is not charging.</p>

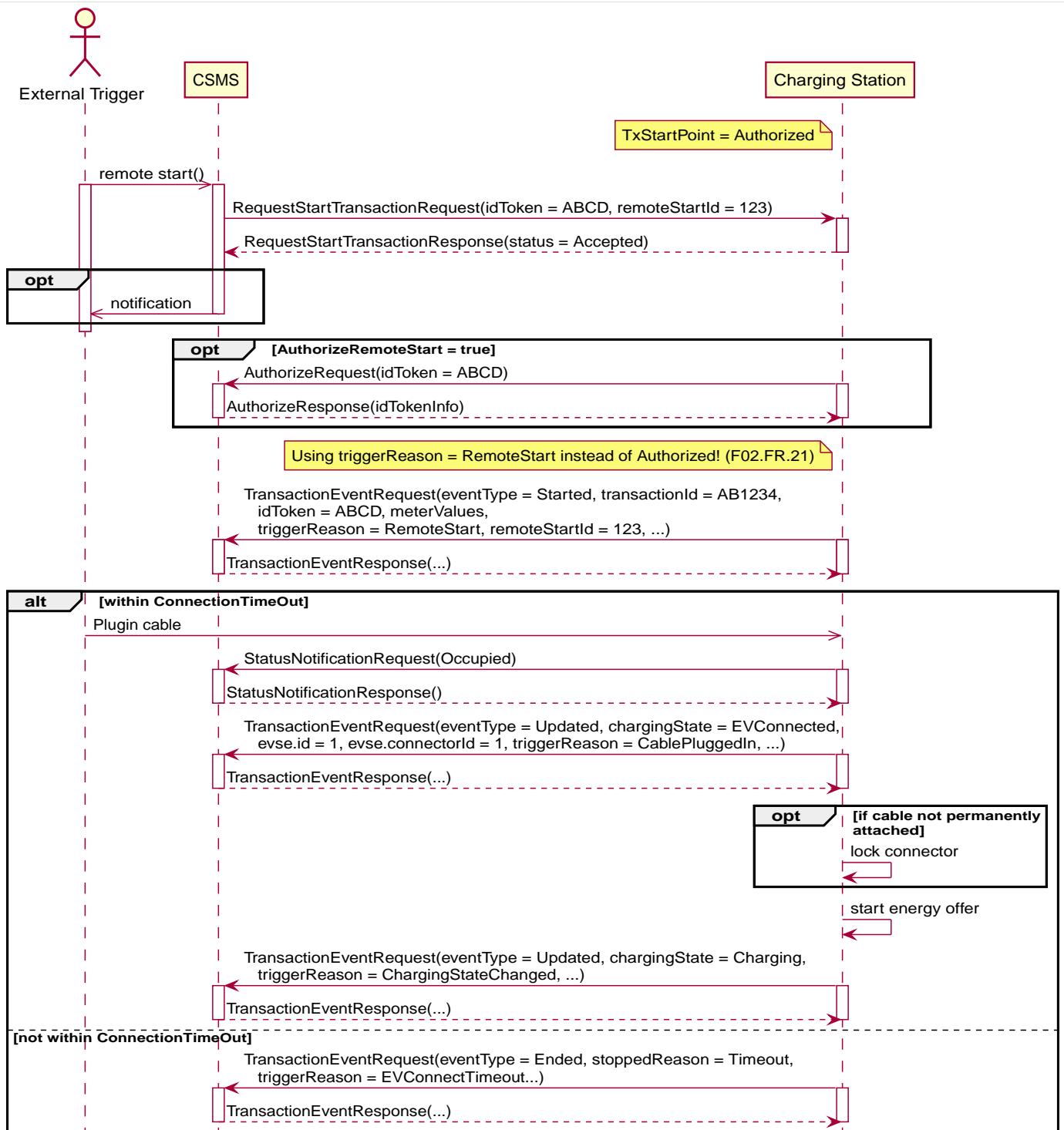


Figure 67. Sequence Diagram: Remote Start Transaction - Remote Start First with TxStartPoint=Authorized

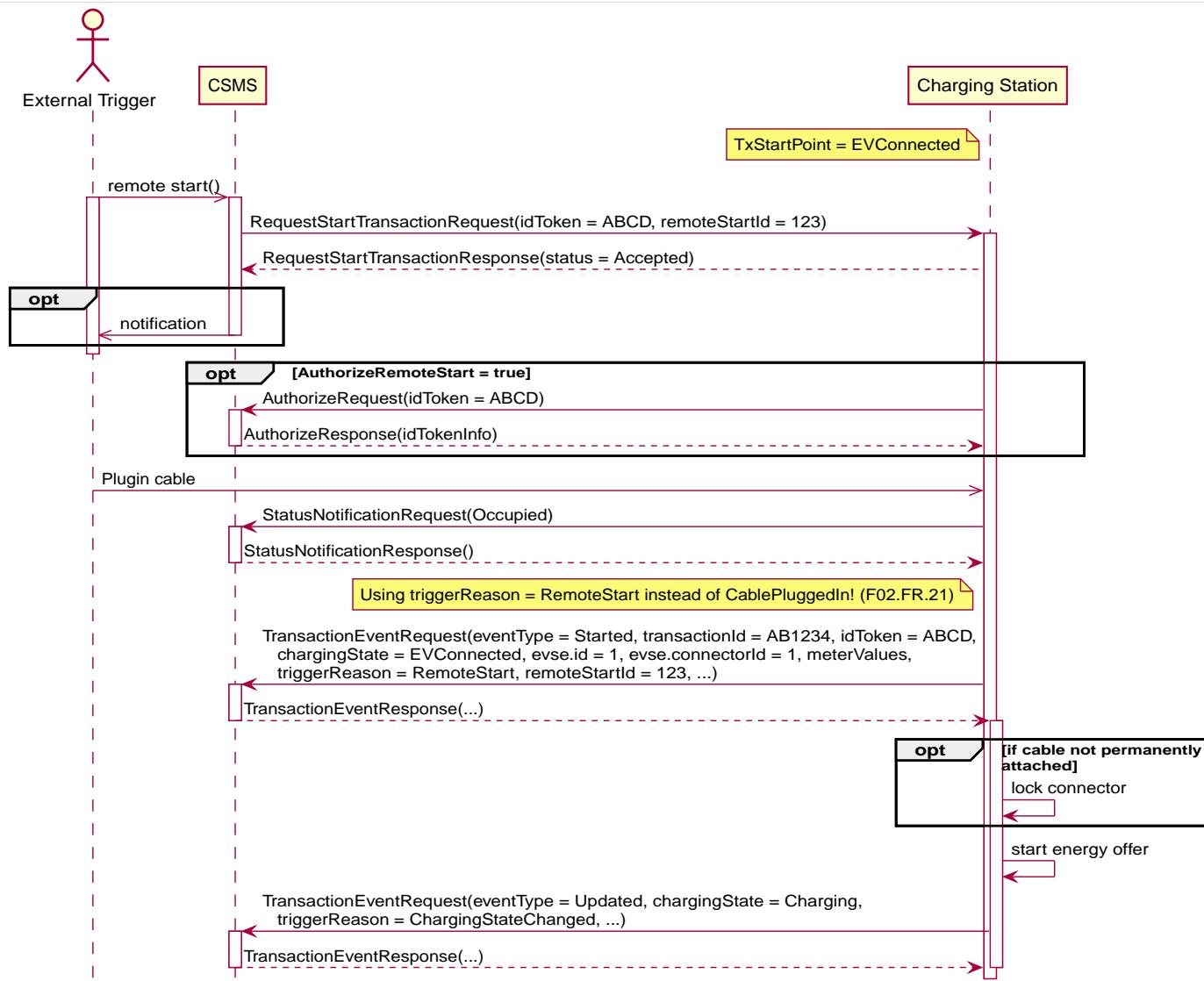


Figure 68. Sequence Diagram: Remote Start Transaction - Remote Start First with TxStartPoint=EVConnected

7	Error handling	n/a
8	Remark(s)	<p>An external trigger can be e.g. a Charging Station Operator or an EV Driver app.</p> <p>It is advised not to start transactions remotely without evsel due to the uncertainty which EVSE is started. In case of a Logic Controller with many EVSEs, the EV Driver might not be in front of the activated EVSE.</p> <p>The CSMS is allowed to send a RequestStartTransactionRequest with IdTokenType of type: NoAuthorization. The operator should be aware that if the Charging Station supports local stop transaction, this transaction can be stopped by anyone.</p> <p>The scenario description and sequence diagram above are based on the Configuration Variable for start transaction being configured as follows: TxStartPoint: EVConnected, Authorized, DataSigned, PowerPathClosed, EnergyTransfer This use-case is also valid for other configurations, but then the transaction might start/stop at another moment, which might change the sequence in which message are send. For more details see the use cases: E01 - Start Transaction options.</p>

F02 - Remote Start Transaction - Remote Start First - Requirements

Table 127. F02 - Requirements

ID	Precondition	Requirement definition	Note
F02.FR.01	When a transaction is started as a result of a RequestStartTransactionRequest .	The Charging Station SHALL put the <i>remoteStartId</i> in the first TransactionEventRequest it sends for this new transaction.	
F02.FR.02	When configured to send meter data in the TransactionEventRequest (<i>eventType</i> = Started), See: Meter Values - Configuration	The Charging Station SHALL add the configured measurands to the optional <i>meterValue</i> field in the TransactionEventRequest (<i>eventType</i> = Started) sent to the CSMS to provide more details during the transaction.	
F02.FR.03	When configured to send meter data in the TransactionEventRequest (<i>eventType</i> = Updated), See: Meter Values - Configuration	The Charging Station SHALL add the configured measurands to the optional <i>meterValue</i> field in the TransactionEventRequest (<i>eventType</i> = Updated) sent to the CSMS to provide more details during the transaction.	
F02.FR.04	F02.FR.03 AND Amount of meter data is too much for 1 TransactionEventRequest (<i>eventType</i> = Updated)	The Charging Station MAY split meter data over multiple TransactionEventRequest (<i>eventType</i> = Updated) messages with the same <i>timestamp</i> .	
F02.FR.05	When the <i>IdToken</i> information is known.	The next TransactionEventRequest SHALL contain <i>IdTokenType</i> information.	
F02.FR.06	This transaction ends a reservation for the specific <i>IdToken</i> .	The next TransactionEventRequest SHALL contain the <i>reservationId</i> .	See H. Reservation .
F02.FR.07	When the EV Driver does not plug-in the charging cable before the timeout set by the Configuration Variable: EVConnectionTimeOut AND TxStopPoint does not contain ParkingBayOccupancy	The Charging Station SHALL end the transaction and send a TransactionEventRequest (<i>eventType</i> = Ended, <i>stoppedReason</i> = Timeout, <i>triggerReason</i> = EVConnectionTimeout) to the CSMS.	Otherwise the transaction would not be ended in case the TxStopPoint does not contain Authorized.
F02.FR.08	When the EV Driver does not plug-in the charging cable before the timeout set by the Configuration Variable: EVConnectionTimeOut AND TxStopPoint contains ParkingBayOccupancy	The Charging Station SHALL deauthorize the transaction and send a TransactionEventRequest (<i>triggerReason</i> = EVConnectionTimeout) to the CSMS.	Transaction will be ended normally when driver leaves the parking bay.
F02.FR.09	If the value of AuthorizeRemoteStart = true AND Charging Station receives a RequestStartTransactionRequest	The Charging Station SHALL behave as if in response to a local action at the Charging Station to start a transaction after successful authorization of the <i>IdToken</i> given in RequestStartTransactionRequest message.	Charging Station will first respond to the request and then try to authorize the <i>IdToken</i> , using the Local Authorization List, Authorization Cache and/or an AuthorizeRequest . A transaction is only started after authorization was obtained.
F02.FR.10	If the value of AuthorizeRemoteStart = false AND Charging Station receives a RequestStartTransactionRequest	The Charging Station SHALL start a transaction for the <i>IdToken</i> given in RequestStartTransactionRequest message without checking authorization.	Note that after the transaction has been started, the Charging Station will send a TransactionEventRequest with the <i>idToken</i> to the CSMS, and the CSMS will check the authorization status of the <i>IdToken</i> when processing this TransactionEventRequest .
F02.FR.11	F02.FR.09 OR F02.FR.10	The Charging Station SHALL send a TransactionEventRequest to the CSMS, and the CSMS will check the authorization status of the <i>IdToken</i> when processing this TransactionEventRequest .	

ID	Precondition	Requirement definition	Note
F02.FR.12		RequestStartTransactionRequest SHALL contain an <i>IdToken</i> , which Charging Station SHALL use, if it is able to start a transaction, in the TransactionEventRequest sent to the CSMS.	
F02.FR.13		The transaction SHALL be started in the same way as described in E03 - Start Transaction - Id Token First .	
F02.FR.14		RequestStartTransactionRequest MAY contain an <i>evseld</i> if the transaction is to be started on a specific EVSE.	When no <i>evseld</i> is provided, the Charging Station is in control of the EVSE selection.
F02.FR.15	If the RequestStartTransactionRequest does not contain an <i>evseld</i> .	The Charging Station MAY reject the RequestStartTransactionRequest .	
F02.FR.16		The CSMS MAY include a ChargingProfile in the RequestStartTransactionRequest .	
F02.FR.17	F02.FR.16	The purpose of this ChargingProfile SHALL be set to TxProfile .	
F02.FR.18	F02.FR.16	The Charging Station SHALL use this ChargingProfile for the transaction that is started by this RequestStartTransaction .	
F02.FR.19	F02.FR.16	The transactionId in the ChargingProfile SHALL NOT be set.	
F02.FR.20	If a Charging Station without support for Smart Charging receives a RequestStartTransactionRequest with a ChargingProfile .	The Charging Station SHALL ignore the specified ChargingProfile .	The device model variable <i>SmartChargingCtrlr.Enabled</i> tells CSMS whether smart charging is supported.
F02.FR.21	When a RequestStartTransactionRequest is received.	The next TransactionEventRequest SHALL contain <i>triggerReason: RemoteStart</i> and the <i>remoteStartId</i> from the RequestStartTransactionRequest .	This is to notify CSMS that this is the result of RequestStartTransaction . Note, that if <i>TxStartPoint=EVConnected</i> the transaction will be started upon cable connection, but the <i>triggerReason = RemoteStart</i> must still be sent. The connection event is reported by the fact that <i>chargingState = EVConnected</i> .
F02.FR.22	If the RequestStartTransactionRequest does not contain an <i>evseld</i> AND the Charging Station is capable of selecting an EVSE	The Charging Station SHALL select an EVSE to be used as a value for <i>evseld</i> for the operation	See also F02.FR.15 if Charging Station does not support starting at an arbitrary EVSE.
F02.FR.23	When the <i>evseld</i> for RequestStartTransactionRequest is Reserved for an <i>idToken</i> that differs from <i>idToken</i> in the request AND has no reservation for a <i>groupIdToken</i>	The Charging Station SHALL respond with RequestStartTransactionResponse with <i>status = Rejected</i> .	
F02.FR.24	When the <i>evseld</i> for RequestStartTransactionRequest is Reserved for an <i>idToken</i> that differs from <i>idToken</i> in the request AND is Reserved for a <i>groupIdToken</i> that differs from <i>groupIdToken</i> in the request	The Charging Station SHALL respond with RequestStartTransactionResponse with <i>status = Rejected</i> .	EV is not allowed to use station if neither <i>idToken</i> nor <i>idGroupToken</i> match the reservation.

ID	Precondition	Requirement definition	Note
F02.FR.25	When the evsId for RequestStartTransactionRequest is Unavailable or Faulted	The Charging Station SHALL respond with RequestStartTransactionResponse with status = Rejected.	
F02.FR.26	When the evsId for RequestStartTransactionRequest is Occupied AND this evsId has a transaction that has been authorized	The Charging Station SHALL respond with RequestStartTransactionResponse with status = Rejected.	Only an EVSE with no transaction or with a transaction that has not yet been authorized can be matched with the RequestStartTransaction Request
F02.FR.27	If a Charging Station with support for Smart Charging receives a RequestStartTransactionRequest with an invalid ChargingProfile .	The Charging Station SHALL respond with RequestStartTransactionResponse with status = Rejected and optionally with <i>reasonCode</i> = "InvalidProfile" or "InvalidSchedule".	The device model variable SmartChargingCtrlr.Enabled tells CSMS whether smart charging is supported.

NOTE

Requirements of previous use case: [F01 - Remote Start Transaction - Cable Plugin First](#), are also considered relevant for [F02 - Remote Start Transaction - Remote Start First](#)

F03 - Remote Stop Transaction

Table 128. F03 - Remote Stop Transaction

No.	Type	Description
1	Name	Remote Stop Transaction
2	ID	F03
	<i>Functional block</i>	F. Remote Control
3	Objective(s)	<ol style="list-style-type: none"> 1. To enable a CSO to help an EV Driver who has problems stopping a transaction. or 2. Enable mobile apps to control transactions via the CSMS.
4	Description	This use case describes how the CSMS requests the Charging Station to stop a transaction.
	<i>Actors</i>	Charging Station, CSMS, CSO, EV Driver
	<i>Scenario description</i>	<ol style="list-style-type: none"> 1. An External Trigger triggers a remote stop. 2. The CSMS requests a Charging Station to stop a transaction by sending RequestStopTransactionRequest to the Charging Station with the transactionId of the transaction. 3. The Charging Station responds with RequestStopTransactionResponse and a status indicating whether it has accepted the request and a transaction with the given transactionId is ongoing and will be stopped. 4. Charging is stopped, the Charging Station sends TransactionEventRequest (<i>eventType</i> = Updated) and, if applicable, unlocks the Connector. 5. After the EV Driver unplugs the cable, the Charging Station sends StatusNotificationRequest with status Available. 6. The Charging Station ends the transaction and sends a TransactionEventRequest (<i>eventType</i> = Ended, <i>stoppedReason</i> = Remote) message to the CSMS.
5	Prerequisite(s)	A transaction is ongoing.
6	Postcondition(s)	Successful postcondition: The transaction for which a stop was requested has ended. Failure postcondition: The transaction for which a stop was requested is still ongoing.

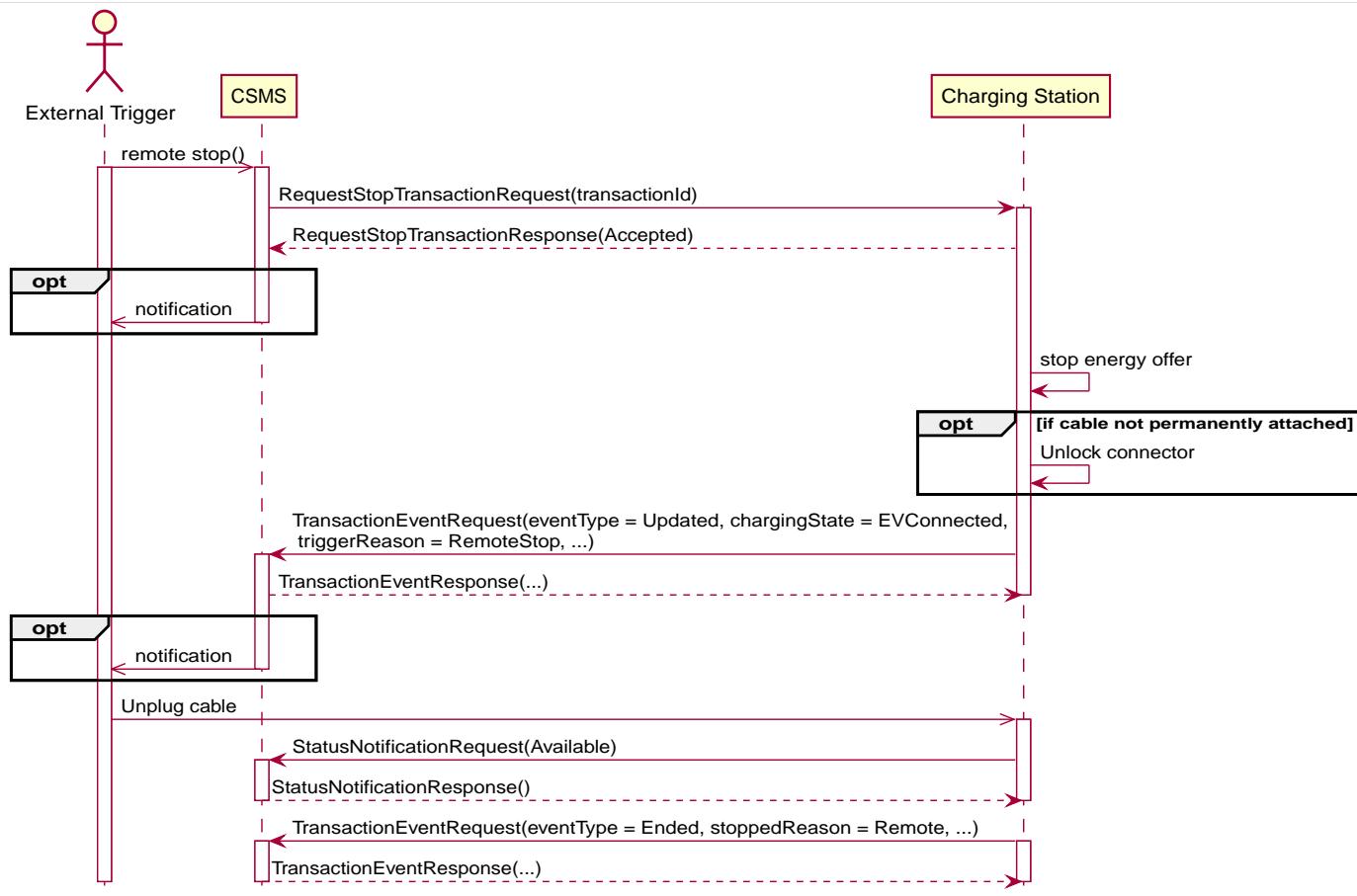


Figure 69. Sequence Diagram: Remote Stop Transaction

7	Remark(s)	<p>This remote request to stop a transaction is equal to a local action to stop a transaction.</p> <p>The scenario description and sequence diagram above are based on the Configuration Variable for stop transaction being configured as follows.</p> <p>TxStopPoint: ParkingBayOccupancy, EVConnected</p> <p>This use-case is also valid for other configurations, but then the transaction might stop at another moment, which might change the sequence in which message are send. For more details see the use case: E06 - Stop Transaction options</p>
---	------------------	--

F03 - Remote Stop Transaction - Requirements

Table 129. F03 - Requirements

ID	Precondition	Requirement definition	Note
F03.FR.01	When the CSMS receives a remote stop transaction trigger (For example when terminating using a smartphone app, exceeding a (non local) prepaid credit.)	The CSMS SHALL send a RequestStopTransactionRequest to the Charging Station with the transactionId of the transaction.	
F03.FR.02	F03.FR.01 AND TxStopPoint configuration does not cause transaction to end (E.g. TxStopPoint is NOT Authorized or PowerPathClosed)	The Charging Station SHALL stop the energy offer and send a TransactionEventRequest (eventType = Updated, triggerReason = RemoteStop) to the CSMS.	For example when TxStopPoint = EVConnected the transaction will not be ended until EV is disconnected.
F03.FR.03	F03.FR.01 AND TxStopPoint configuration causes transaction to end (E.g. TxStopPoint is NOT Authorized or PowerPathClosed)	The Charging Station SHALL send a TransactionEventRequest (eventType = Ended, triggerReason = RemoteStop, stoppedReason = Remote) to the CSMS.	For example when TxStopPoint = EVConnected and EV is disconnected after the RequestStopTransaction Request.

ID	Precondition	Requirement definition	Note
F03.FR.04	When configured to send meter data in the TransactionEventRequest (<code>eventType = Ended</code>), See: Meter Values - Configuration	The Charging Station SHALL add the configured measurands to the optional <code>meterValue</code> field in the TransactionEventRequest (<code>eventType = Ended</code>) sent to the CSMS to provide more details about transaction usage.	
F03.FR.05	F03.FR.04 AND The Charging Station is running low on memory	The Charging Station MAY drop meter data.	
F03.FR.06	F03.FR.05	When dropping meter data, the Charging Station SHALL drop intermediate values first (1st value, 3th value, 5th etc), not start dropping values from the start of the list or stop adding values to the list.	
F03.FR.07	When the Charging Station receives a RequestStopTransactionRequest	And the TransactionId can be matched to an active transaction; the Charging Station SHALL respond with a RequestStopTransactionResponse with status set to <code>Accepted</code> .	
F03.FR.08	When the Charging Station receives a RequestStopTransactionRequest	And the TransactionId cannot be matched to an active transaction; the Charging Station SHALL respond with a RequestStopTransactionResponse with status set to <code>Rejected</code> .	
F03.FR.09	When sending a TransactionEventRequest	The Charging Station SHALL set the <code>triggerReason</code> to inform the CSMS about what triggered the event. What reason to use is described in the description of TriggerReasonEnumType .	

F04 - Remote Stop ISO 15118 Charging from CSMS

Table 130. F04 - Charging loop with interrupt from the CSMS

No.	Type	Description
1	Name	Remote Stop ISO 15118 Charging from CSMS
2	ID	F04
	<i>Functional block</i>	F. Remote Control
	<i>Reference</i>	ISO15118-1 F2 Charging loop with interrupt from the SECC.
3	Objectives	See ISO15118-1 , use case Objective F2, page 38.
4	Description	See ISO15118-1 , use case Description F2, page 38.
5	Actors	EV, EVSE, Charging Station
6	Prerequisites	- If authorization according use cases in Functional Block C is applied, it SHALL be finished successfully. See ISO15118-1 , use case Prerequisites F2, page 38.
7	Combined scenario description	<p>OCPP:</p> <ol style="list-style-type: none"> 1. The CSMS sends a RequestStopTransactionRequest to the Charging Station. 2. The Charging Station responds with a RequestStopTransactionResponse. <p>ISO 15118:</p> <ol style="list-style-type: none"> 3. The EV sends a <code>ChargingStatus</code> (in case of AC charging) or <code>CurrentDemandReq</code> (in case of DC Charging) PDU to the Charging Station. 4. The Charging Station responds with an <code>EVSENNotification = StopCharging</code>.
8	Postcondition(s)	See ISO15118-1 , use case End conditions F2, page 38.

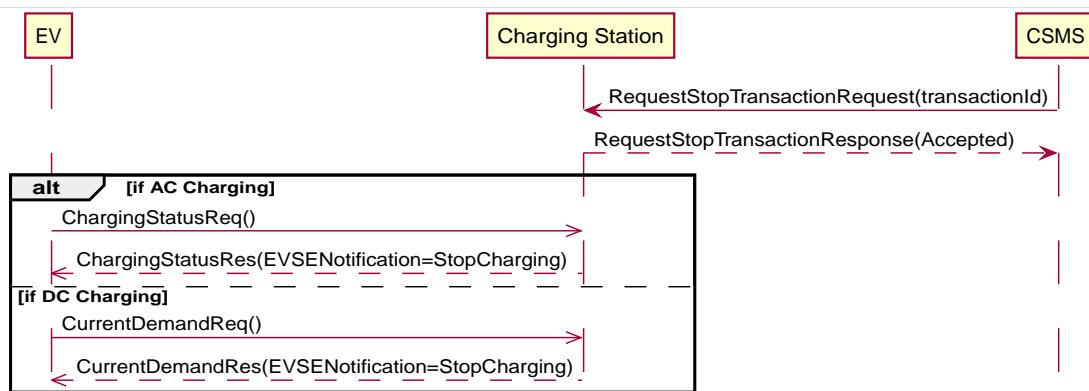


Figure 70. Charging loop with interrupt from the Charging Station

9	Error handling	n/a
10	Remark(s)	n/a

F04 - Remote Stop ISO 15118 Charging from CSMS - Requirements

These requirements are normative.

Table 131. F04 - Requirements

ID	Precondition	Requirement definition	Note
F04.FR.01	When the CSMS receives a remote stop transaction trigger (For example when terminating using a smartphone app, exceeding a (non local) prepaid credit.)	The CSMS SHALL send a RequestStopTransactionRequest to the Charging Station with the transactionId of the transaction.	
F04.FR.02	F04.FR.01	The Charging Station SHALL stop the energy offer, unlock the cable and send a TransactionEventRequest (eventType = Updated) to the CSMS.	Cable unlocked if not permanently attached.
F04.FR.03	F04.FR.02 AND When the EV Driver unplugs the cable.	The Charging Station SHALL send a TransactionEventRequest (eventType = Ended , stoppedReason = Remote) to the CSMS.	
F04.FR.04	When configured to send meter data in the TransactionEventRequest (eventType = Ended), See: Meter Values - Configuration	The Charging Station SHALL add the configured measurands to the optional meterValue field in the TransactionEventRequest (eventType = Ended) sent to the CSMS to provide more details about transaction usage.	
F04.FR.05	F04.FR.04 AND The Charging Station is running low on memory	The Charging Station MAY drop meter data.	
F04.FR.06	F04.FR.05	When dropping meter data, the Charging Station SHALL drop intermediate values first (1st value, 3th value, 5th etc), not start dropping values from the start of the list or stop adding values to the list.	

2.2. Unlock Connector

F05 - Remotely Unlock Connector

Table 132. F05 - Remotely Unlock Connector

No.	Type	Description
1	Name	Remotely Unlock Connector
2	ID	F05
	Functional block	F. RemoteControl
3	Objective(s)	To enable the CSO to help an EV-driver that has problems unplugging his charging cable because the lock failed after the transaction has ended.
4	Description	It sometimes happens that a connector of a Charging Station socket does not unlock correctly. This happens most of the time when there is tension on the charging cable. This means the driver cannot unplug his charging cable from the Charging Station. To help a driver, the CSO can send a UnlockConnectorRequest to the Charging Station. The Charging Station will then try to unlock the connector again.
	Actors	Charging Station, CSMS, External Trigger
	Scenario description	<ol style="list-style-type: none"> An External Trigger (probably the CSO) request the unlocking of a specific connector of a Charging Station. The CSMS sends an UnlockConnectorRequest to the Charging Station. Upon receipt of UnlockConnectorRequest, the Charging Station responds with UnlockConnectorResponse. The response message indicates whether the Charging Station was able to unlock its Connector.
5	Prerequisite(s)	No ongoing transaction on the specified connector The Charging Station has a connector lock.
6	Postcondition(s)	The Charging Station was able to unlock the Connector.

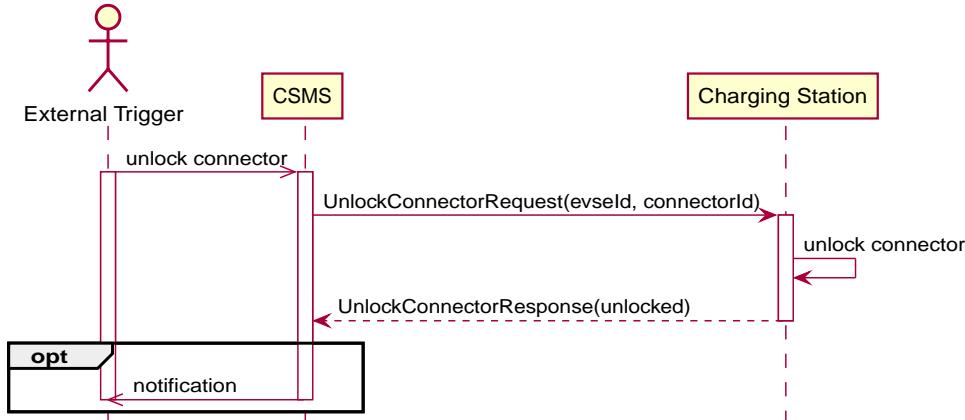


Figure 71. Sequence Diagram: Unlock Connector

7	Error handling	n/a
8	Remark(s)	<p>An external trigger, triggering the Unlock command, can be e.g. a Charging Station Operator or an EV Driver app.</p> <p>UnlockConnectorRequest is intended only for unlocking the cable retention lock on the Connector, not for unlocking a Connector access door.</p>

F05 - Remotely Unlock Connector - Requirements

Table 133. F05 - Requirements

ID	Precondition	Requirement definition
F05.FR.01	Upon receipt of an UnlockConnectorRequest .	The Charging Station SHALL respond with UnlockConnectorResponse .

ID	Precondition	Requirement definition
F05.FR.02	F05.FR.01 AND There is a an authorized transaction ongoing on the specified connector.	The Charging Station SHALL NOT try to unlock the connector (or stop the transaction) but use the status: OngoingAuthorizedTransaction in the UnlockConnectorResponse .
F05.FR.03	F05.FR.01 AND Specified connector unknown.	The Charging Station SHALL use the status: UnknownConnector in the UnlockConnectorResponse .
F05.FR.04	F05.FR.01 AND The Charging Station was able to unlock the specified connector.	The Charging Station SHALL use the status: Unlocked in the UnlockConnectorResponse .
F05.FR.05	F05.FR.01 AND The Charging Station was NOT able to unlock the specified connector.	The Charging Station SHALL use the status: UnlockFailed in the UnlockConnectorResponse .
F05.FR.06	F05.FR.01 AND No cable is connected to the connector.	The Charging Station SHALL attempt to unlock the connector, even if no cable is detected and SHALL return the result of the unlock attempt.

2.3. Remote Trigger

F06 - Trigger Message

Table 134. F06 - Trigger Message

No.	Type	Description
1	Name	Trigger Message
2	ID	F06
	Functional block	F. RemoteControl
3	Objective(s)	To enable the CSMS to request a Charging Station to send a Charging Station-initiated message.
4	Description	This use case describes the use of the TriggerMessageRequest message: how a CSMS can request a Charging Station to send Charging Station-initiated messages. In the request the CSMS indicates which message it wishes to receive.
	Actors	Charging Station, CSMS
	Scenario description	<p>1. The CSMS sends a TriggerMessageRequest to the Charging Station.</p> <p>2. The Charging Station responds with a TriggerMessageResponse, indicating whether it will send it or not, by returning Accepted, Rejected or NotImplemented.</p> <p>3. Message, requested by the CSMS, that the Charging Station marked as Accepted, is being sent.</p>
5	Prerequisite(s)	The Functional Block <i>Remote Trigger</i> is installed.
6	Postcondition(s)	<p>Successful postconditions:</p> <p>1. The CSMS has Successfully received a TriggerMessageResponse message.</p> <p>2. The CSMS has Successfully received a TriggerMessageResponse message with status Accepted AND has Successfully received the requested message.</p> <p>Failure postconditions:</p> <p>1. The CSMS has NOT received a TriggerMessageResponse message.</p> <p>2. The CSMS has Successfully received a TriggerMessageResponse message with status Accepted AND has NOT received the requested message.</p>

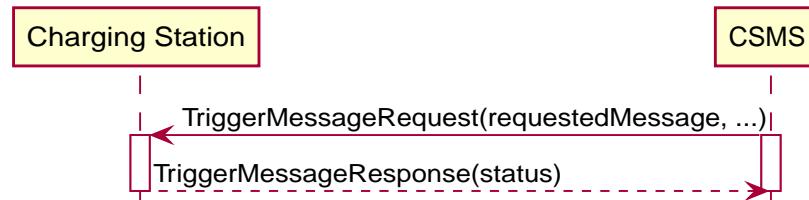


Figure 72. Sequence Diagram: Trigger Message

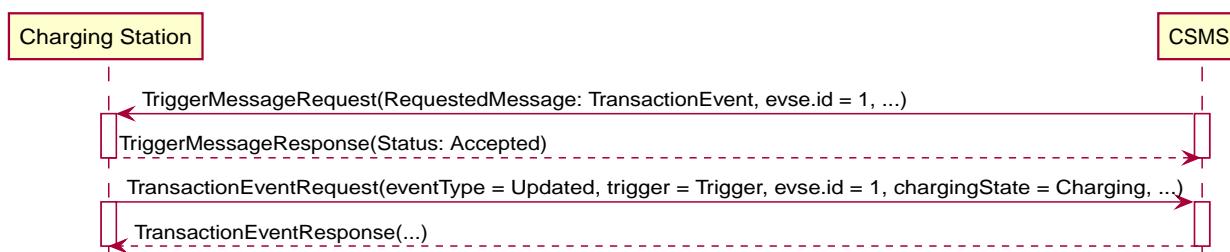


Figure 73. Sequence Diagram: Trigger Message Example

7	Error handling	n/a
8	Remark(s)	The TriggerMessage mechanism is not intended to retrieve historic data.

F06 - Trigger Message - Requirements

Table 135. F06 - Requirements

ID	Precondition	Requirement definition	Note
F06.FR.01		In the TriggerMessageRequest message, the CSMS SHALL indicate which message(s) it wishes to receive.	
F06.FR.02	F06.FR.01. For every such requested message.	The CSMS MAY indicate to which EVSE this request applies.	
F06.FR.03	F06.FR.02	The requested message SHALL be leading. If the specified evsel is not relevant to the message, it SHALL be ignored. In such cases the requested message SHALL still be sent.	
F06.FR.04	If a Charging Station receives a TriggerMessageRequest .	The Charging Station SHALL first send the TriggerMessageResponse , before sending the requested message.	
F06.FR.05	F06.FR.04	In the TriggerMessageResponse the Charging Station SHALL indicate whether it will send the requested message or not, by returning Accepted or Rejected.	It is up to the Charging Station if it accepts or rejects the request to send.
F06.FR.06	If a Charging Station receives a TriggerMessageRequest with requestedMessage set to: MeterValues	The Charging Station SHALL send a MeterValuesRequest to the CSMS with the most recent measurements for all measurands configured in Configuration Variable: AlignedDataMeasurands .	
F06.FR.07	If a Charging Station receives a TriggerMessageRequest with requestedMessage set to: TransactionEvent	The Charging Station SHALL send a TransactionEventRequest to the CSMS with triggerReason = Trigger, transactionInfo with at least the chargingState, and meterValue with the most recent measurements for all measurands configured in Configuration Variable: SampledDataTxUpdatedMeasurands .	
F06.FR.08	When the Charging Station receives a TriggerMessageRequest with a requestedMessage that it has not implemented	The Charging Station SHALL respond with TriggerMessageResponse with status NotImplemented.	
F06.FR.09		The messages it triggers SHALL only give current information.	
F06.FR.10		Messages that the Charging Station marks as Accepted SHALL be sent.	E.g. the situation could occur that, between accepting the request and actually sending the requested message, that same message gets sent because of normal operations. In such cases the message just sent MAY be considered as complying with the request.
F06.FR.11	If the field evse is relevant but absent in the TriggerMessageRequest .	The Charging Station SHALL interpret this as "for all allowed evse values".	StatusNotifications can only be requested for a specific connector, see F06.FR.12/13
F06.FR.12	If a Charging Station receives a TriggerMessageRequest with requestedMessage set to: StatusNotification AND (evse is omitted OR evse.connectorId is omitted)	The Charging Station SHALL respond with a TriggerMessageResponse with status Rejected.	StatusNotification messages can only be sent at connector level.

ID	Precondition	Requirement definition	Note
F06.FR.13	When sending a TriggerMessageRequest with <i>requestedMessage</i> set to: <i>StatusNotification</i>	The CSMS SHALL set the connectorId field	StatusNotification messages can only be sent at connector level.
F06.FR.14	If a Charging Station receives a TriggerMessageRequest with <i>requestedMessage</i> set to: <i>LogStatusNotification</i> AND The Charging Station is uploading a log file	The Charging Station SHALL send a LogStatusNotificationRequest to the CSMS with <i>status Uploading</i> .	
F06.FR.15	If a Charging Station receives a TriggerMessageRequest with <i>requestedMessage</i> set to: <i>LogStatusNotification</i> AND The Charging Station is NOT uploading a log file	The Charging Station SHALL send a LogStatusNotificationRequest to the CSMS with <i>status Idle</i> .	
F06.FR.16	If a Charging Station receives a TriggerMessageRequest with <i>requestedMessage</i> set to: <i>FirmwareStatusNotification</i> AND The Charging Station is not performing firmware update related tasks.	The Charging Station SHALL send a FirmwareStatusNotificationRequest to the CSMS with <i>status Idle</i> .	
F06.FR.17	If Charging Station receives a TriggerMessageRequest with <i>requestedMessage</i> set to: <i>BootNotification</i> AND the response it received from CSMS to the last BootNotificationRequest was: <i>Accepted</i>	Charging Station SHALL respond with a TriggerMessageResponse with status <i>Rejected</i> .	A trigger to request a Charging Station to send a BootNotification is only meant to be used when the BootNotification has not yet been accepted.

G. Availability

1. Introduction

This Functional Block specifies how the Charging Station can inform the CSMS of its current availability for starting new transactions.

For the CSO it is important to know if a Charging Station is available for new EVs to be charged. The CSO wants to know this information so they can tell EV Drivers whether the Charging Station is available. To know this, the Charging Station should send any status changes of itself or one of its EVSEs to the CSMS. See for an example: [B04 - Offline Behavior Idle Charging Station](#).

For the CSO it is very helpful to know the status of the transaction, therefore the Charging Station can send detailed statuses to the CSMS. This can be very useful when helping an EV Driver when he experiences problems during charging.

When a fault is detected by the Charging Station it can send a message notifying the CSMS about the fault.

When the CSO wants the Charging Station to no longer start new transactions, it can change the availability. For example: they need to do maintenance on the Charging Station, and for this reason they don't want the Charging Station to be in use.

The CSO can also change the availability for one or more EVSEs. For example: A customer calls, complaining about a broken EVSE on the Charging Station. The CSO can then set the Connector to unavailable, making it impossible for an EV Driver to use that Connector.

Obviously, it is also possible to make the Charging Station or a Connector available again with a command from the CSMS.

NOTE

An overview of the Connectors Statuses can be found in: [ConnectorStatusEnumType](#).

2. Use cases & Requirements

G01 - Status Notification

Table 136. G01 - Status Notification

No.	Type	Description
1	Name	Status Notification
2	ID	G01
	Functional block	G. Availability
3	Objective(s)	To inform the CSMS about a Connector status change.
4	Description	This use case covers the functionality that a Charging Station sends a notification to the CSMS to inform the CSMS about a Connector status change.
	Actors	Charging Station, CSMS
	Scenario description	<p>1. A connector status changed, the Charging Station sends a StatusNotificationRequest to the CSMS to inform the CSMS about the new status.</p> <p>2. The CSMS responds with StatusNotificationResponse to the Charging Station.</p>
5	Prerequisite(s)	n/a
6	Postcondition(s)	<p>Successful postconditions: The CSMS is Successfully informed about the status change.</p> <p>Failure postconditions: n/a</p>

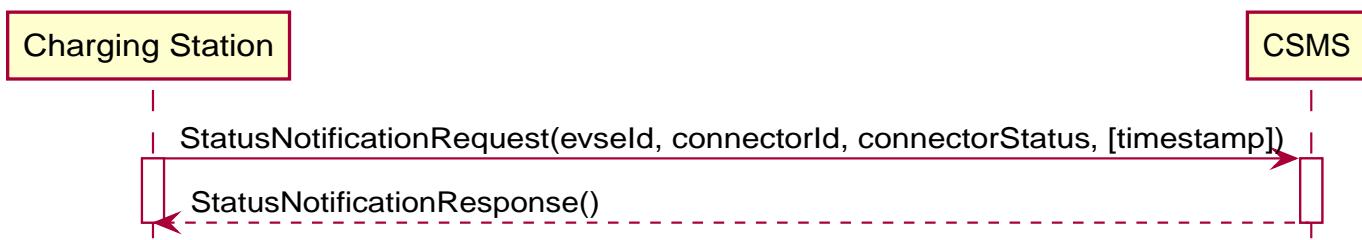


Figure 74. Sequence Diagram: Status Notification

7	Error handling	n/a
8	Remark(s)	<p>The Charging Station MAY use the <i>Unavailable</i> status internally for other purposes (e.g. while updating firmware or waiting for an initial Accepted RegistrationStatus). When one of the connectors on an EVSE is Reserved/Occupied, the CSMS has to take care of the status of the other connectors when presenting availability information to another system or user. The CSMS knows which connectors belong to the same EVSE.</p> <p>Notifying a connector status from the Charging Station to the CSMS will be taken over by the new Device Management Monitoring feature, however this mechanism has not been proven in the field yet. So the old StatusNotificationRequest message remains available for use for now.</p>

G01 - Status Notification - State transition overview for connecting/disconnecting

Initial	Cable plugin	Cable unplug
Available	→ Occupied	-

Initial	Cable plugin	Cable unplug
Occupied	-	→ Available (→ Unavailable, if scheduled to become Unavailable)
Reserved	(→ Occupied, only if authorized for reserved IdToken)	-
Unavailable	-	-
Faulted	-	-

G01 - Status Notification - Requirements

Table 137. G01 - Requirements

ID	Precondition	Requirement definition
G01.FR.01		A Charging Station Connector MUST have one of the valid statuses from the ConnectorStatus enumeration.
G01.FR.02	When an EVSE is set to status <i>Unavailable</i> by a ChangeAvailabilityRequest message.	The EVSE's <i>Unavailable</i> status SHALL be persistent across reboots.
G01.FR.03	The connector is <i>Available</i> when an EV is connecting	The Charging Station SHALL send a StatusNotificationRequest with <i>connectorStatus</i> = <i>Occupied</i> or a NotifyEventRequest for <i>component</i> = "Connector", <i>variable</i> = "AvailabilityState", <i>actualValue</i> = "Occupied" and <i>trigger</i> = "Delta".
G01.FR.04	The connector is <i>Occupied</i> when an EV is disconnecting AND connector is not scheduled to become <i>Unavailable</i> (G03.FR.05)	The Charging Station SHALL send a StatusNotificationRequest with <i>connectorStatus</i> = <i>Available</i> when an EV is disconnected or a NotifyEventRequest for <i>component</i> = "Connector", <i>variable</i> = "AvailabilityState", <i>actualValue</i> = "Available" and <i>trigger</i> = "Delta".
G01.FR.05	The connector is <i>Occupied</i> when an EV is disconnecting AND connector is scheduled to become <i>Unavailable</i> (G03.FR.05)	The Charging Station SHALL send a StatusNotificationRequest with <i>connectorStatus</i> = <i>Unavailable</i> when an EV is disconnected or a NotifyEventRequest for <i>component</i> = "Connector", <i>variable</i> = "AvailabilityState", <i>actualValue</i> = "Unavailable" and <i>trigger</i> = "Delta".
G01.FR.06	The connector is <i>Reserved</i> when an EV is connecting AND EV driver presents an IdToken matching the reservation	The Charging Station SHALL send a StatusNotificationRequest with <i>connectorStatus</i> = <i>Occupied</i> or a NotifyEventRequest for <i>component</i> = "Connector", <i>variable</i> = "AvailabilityState", <i>actualValue</i> = "Occupied" and <i>trigger</i> = "Delta".
G01.FR.07	When a ChangeAvailabilityRequest leads to a connector status change	The Charging Station SHALL send a StatusNotificationRequest with the corresponding <i>connectorStatus</i> or a NotifyEventRequest for <i>component</i> = "Connector", <i>variable</i> = "AvailabilityState", <i>trigger</i> = "Delta" and the corresponding <i>actualValue</i> of "AvailabilityState".
G01.FR.08	When a connector of an EVSE becomes reserved or a cable is plugged-in AND The EVSE has multiple connectors	The Charging Station SHOULD NOT send a StatusNotificationRequest for the other connector(s), even though they are no longer usable.

G02 - Heartbeat

Table 138. G02 - Heartbeat

No.	Type	Description
1	Name	Heartbeat
2	ID	G02
	Functional block	G. Availability
3	Objective(s)	To let the CSMS know that a Charging Station is still connected, optionally the Heartbeat can be used for time synchronisation.
4	Description	This use case describes a way to let the CSMS know the Charging Station is still connected, a Charging Station sends a heartbeat after a configurable time interval. Depending on the configuration the Heartbeat can be used for time synchronisation.
	Actors	Charging Station, CSMS
	Scenario description	<p>1. If there is no activity for a certain time, the Charging Station sends HeartbeatRequest for ensuring that the CSMS knows that a Charging Station is still alive.</p> <p>2. Upon receipt of HeartbeatRequest, the CSMS responds with HeartbeatResponse. The response message contains the current time of the CSMS, which the Charging Station MAY use to synchronize its internal clock.</p>
5	Prerequisite(s)	The heartbeat interval is set.
6	Postcondition(s)	<p>Successful postconditions: The CSMS knows the Charging Station is still connected.</p> <p>Failure postconditions: The CSMS concludes that the Charging Station is <i>Offline</i>.</p>

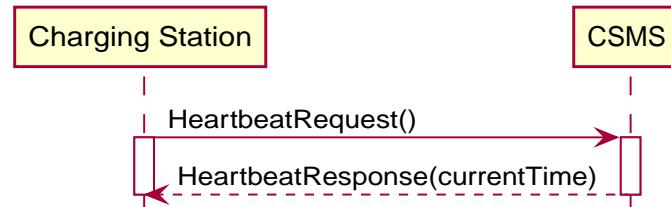


Figure 75. Sequence Diagram: Heartbeat

7	Error handling	n/a
8	Remark(s)	With JSON over WebSocket, sending heartbeats is <i>not</i> instrumental to keeping websockets alive, since websockets already provide a mechanism for this. However, if the Charging Station uses the heartbeat for time synchronization, it is advised to at least send one heartbeat per 24 hours.

G02 - Heartbeat - Requirements

Table 139. G02 - Requirements

ID	Precondition	Requirement definition	Note
G02.FR.01	When the CSMS responds with BootNotificationResponse with a status Accepted.	The Charging Station SHALL adjust the heartbeat interval in accordance with the interval from the response message.	
G02.FR.02		The Charging Station SHALL send HeartbeatRequest after a configurable time interval.	To ensure that the CSMS knows that a Charging Station is still alive.
G02.FR.03		The HeartbeatResponse message SHALL contain the current time of the CSMS.	
G02.FR.04	Whenever a message from a Charging Station has been received.	The CSMS SHALL assume availability of that Charging Station.	
G02.FR.05		It is RECOMMENDED that the Charging Station resets its heartbeat interval timer when another message has been sent to the CSMS.	

ID	Precondition	Requirement definition	Note
G02.FR.06	When the Charging Station receives a HeartbeatResponse .	It is RECOMMENDED that the Charging Station uses the current time to synchronize its internal clock.	
G02.FR.07	When the heartbeat interval timer is continuously reset because of continuous sending of messages AND HeartbeatRequest is used for time synchronisation	It is RECOMMENDED that the Charging Station sends a HeartbeatRequest at least once every 24 hours to synchronise the clock.	

G03 - Change Availability EVSE/Connector

Table 140. G03 - Change Availability EVSE/Connector

No.	Type	Description
1	Name	Change Availability EVSE/Connector
2	ID	G03
	Functional block	G. Availability
3	Objective(s)	To enable the CSMS to change the availability of an EVSE or Connector to <i>Operative</i> or <i>Inoperative</i> .
4	Description	This use case covers how the CSMS requests the Charging Station to change the availability of one of the EVSEs or Connectors to <i>Operative</i> or <i>Inoperative</i> . An EVSE/Connector is considered <i>Operative</i> in any status other than <i>Faulted</i> and <i>Unavailable</i> .
	Actors	Charging Station, CSMS
	Scenario description	<p>1. The CSMS sends ChangeAvailabilityRequest requesting a Charging Station to change the availability of an EVSE or Connector.</p> <p>2. The Charging Station changes the availability to the EVSE/Connector to the requested operationalStatus from the ChangeAvailabilityRequest.</p> <p>3. Upon receipt of ChangeAvailabilityRequest, the Charging Station responds with ChangeAvailabilityResponse. In case that the status 'Scheduled' is reported in the ChangeAvailabilityResponse, a transaction was running and this will be finished first.</p> <p>4. The Charging Station reports the status of the EVSE/Connector using a StatusNotification.</p>
	Alternative scenario(s)	G04 - Change Availability Charging Station
5	Prerequisite(s)	n/a
6	Postcondition(s)	<p>Successful postcondition: When changing the availability of an EVSE/Connector to <i>Operative</i>, the status of the EVSE has changed to <i>Available</i>, <i>Occupied</i> or <i>Reserved</i>. When changing the availability of an EVSE/Connector to <i>Inoperative</i>, the status of the EVSE has changed to <i>Unavailable</i>.</p> <p>Failure postcondition: The status of the EVSE is as it was just before the Charging Station received ChangeAvailabilityRequest and not according to the requested Availability.</p>

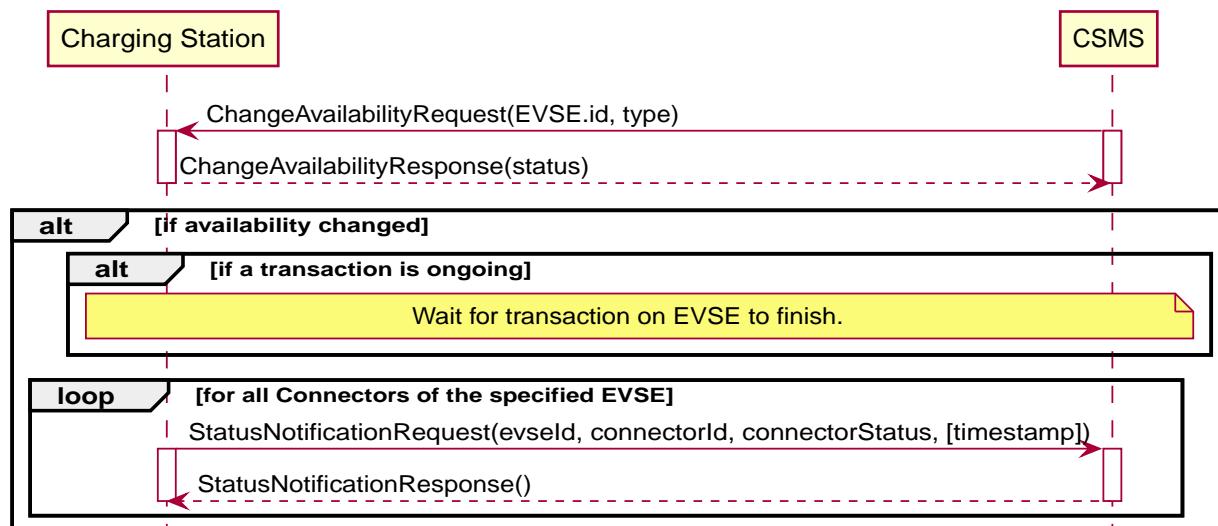


Figure 76. Sequence Diagram: Change Availability

7	Error handling	n/a
8	Remark(s)	Persistent states, for example: EVSE set to <i>Available</i> SHALL persist a reboot.

G03 - Change Availability EVSE - Requirements

Table 141. G03 - Requirements

ID	Precondition	Requirement definition	Note
G03.FR.01	Upon receipt of ChangeAvailabilityRequest .	The Charging Station SHALL respond with ChangeAvailabilityResponse .	
G03.FR.02	G03.FR.01	This response message SHALL indicate whether the Charging Station is able to change to the requested availability.	
G03.FR.03	In the event that CSMS requests the Charging Station to change an EVSE to the state it is already in.	The Charging Station SHALL respond with availability status <i>Accepted</i> .	
G03.FR.04	When an availability change request with ChangeAvailabilityRequest has happened.	The Charging Station SHALL inform the CSMS of its new availability status with StatusNotificationRequest .	As described in ChangeAvailabilityStatusEnumType
G03.FR.05	When a transaction is in progress AND NOT G03.FR.03	The Charging Station SHALL respond with availability status <i>Scheduled</i> to indicate that it is scheduled to occur after the transaction has finished.	
G03.FR.06	When the availability of an EVSE becomes Inoperative (<i>Unavailable</i> , <i>Faulted</i>)	All operative connectors (i.e. not <i>Faulted</i>) of that EVSE SHALL become <i>Unavailable</i> .	
G03.FR.07	When the availability of an EVSE becomes Operative	The Charging Station SHALL revert the status of all connectors of that EVSE to their original status.	See Note 1.
G03.FR.08	When the availability of an EVSE or Connector has been set explicitly via ChangeAvailabilityRequest	The set availability state SHALL be persistent across reboot/power loss.	

NOTE

1. The Charging Station, EVSEs and Connectors have separate / individual states. This means (for example) that when setting a connector to Inoperative, then setting the connected EVSE to Inoperative and thereafter change the EVSE back to operative, the connector will remain Inoperative.

NOTE

2. It is only required to report a status change of a connector. StatusNotificationRequest only supports the reporting of connector statuses.

G04 - Change Availability Charging Station

Table 142. G04 - Change Availability Charging Station

No.	Type	Description
1	Name	Change Availability Charging Station
2	ID	G04
	Functional block	G. Availability
	Parent use case	G03 - Change Availability EVSE/Connector
3	Objective(s)	To enable the CSMS to change the availability of a Charging Station.
4	Description	<p>This use case describes how the CSMS requests the Charging Station to change the availability.</p> <p>A Charging Station is considered <i>Operative</i> when it is charging or ready for charging.</p> <p>A Charging Station is considered <i>Inoperative</i> when it does <i>not</i> allow any charging.</p>
	Actors	Charging Station, CSMS
	Scenario description	<ol style="list-style-type: none"> The CSMS sends a ChangeAvailabilityRequest for requesting a Charging Station to change its availability. Upon receipt of a ChangeAvailabilityRequest, the Charging Station responds with ChangeAvailabilityResponse.
5	Prerequisite(s)	n/a
6	Postcondition(s)	<p>Successful postcondition: The CSMS was able to change the availability of the Charging Station. When changing the availability of a Charging Station to <i>Operative</i>, the status of the Charging Station has changed to <i>Available</i>. When changing the availability of a Charging Station to <i>Inoperative</i>, the status of the Charging Station has changed to <i>Unavailable</i>.</p> <p>Failure postcondition: The CSMS was <i>not</i> able to change the requested Charging Station's availability.</p>

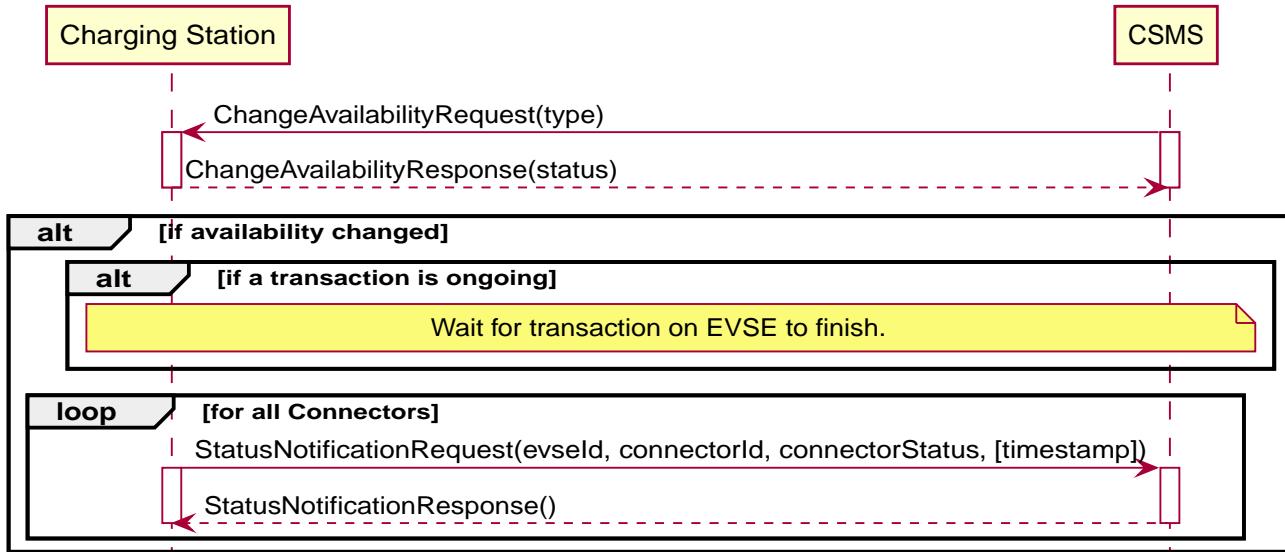


Figure 77. Sequence Diagram: Change Availability Charging Station

7	Error handling	n/a
8	Remark(s)	Persistent states: for example, Charging Station set to <i>Unavailable</i> SHALL persist a reboot.

G04 - Change Availability Charging Station - Requirements

Table 143. G04 - Requirements

ID	Precondition	Requirement definition	Note
G04.FR.01	In the case the evse field is omitted in ChangeAvailabilityRequest .	The Charging Station status change SHALL apply to the whole Charging Station.	
G04.FR.02	Upon receipt of ChangeAvailabilityRequest .	The Charging Station SHALL respond with ChangeAvailabilityResponse .	
G04.FR.03	G04.FR.02	This response message SHALL indicate whether the Charging Station is able to change to the requested availability.	
G04.FR.04	In the event that CSMS requests the Charging Station to change to the state it is already in.	The Charging Station SHALL respond with availability status Accepted.	
G04.FR.05	When an availability change request with ChangeAvailabilityRequest has happened.	The Charging Station SHALL inform the CSMS by sending the status of each of the changed connectors via a StatusNotificationRequest	As described in ConnectorStatusEnumType
G04.FR.06	When a transaction is in progress.	The Charging Station SHALL respond with availability status Scheduled to indicate that it is scheduled to occur after the transaction has finished.	
G04.FR.07	When the availability of the Charging Station becomes Inoperative (<i>Unavailable, Faulted</i>)	All operative EVSEs and connectors (i.e. not <i>Faulted</i>) SHALL become <i>Unavailable</i> .	
G04.FR.08	When the availability of the Charging Station becomes Operative	The Charging Station SHALL revert the status of all EVSEs and connectors to their original status.	See Note 1.
G04.FR.09	When the availability of a Charging Station has been set explicitly via ChangeAvailabilityRequest	The set availability state SHALL be persistent across reboot/power loss.	

NOTE

1. The Charging Station, EVSEs and Connectors have separate / individual states. This means (for example) that when setting a connector to Inoperative, then setting the connected EVSE to Inoperative and thereafter change the EVSE back to operative, the connector will remain Inoperative.

NOTE

2. It is only required to report a status change of a connector. StatusNotificationRequest only supports the reporting of connector statuses.

G05 - Lock Failure

Table 144. G05 - Lock Failure

No.	Type	Description
1	Name	Lock Failure
2	ID	G05
	Functional block	G. Availability
3	Objective(s)	To prevent the EV Driver from charging while the Connector is not properly locked.
4	Description	This use case describes how the EV Driver is prevented from starting a charge session at the Charging Station while the Connector is not locked properly.
	Actors	Charging Station, CSMS, EV Driver
	Scenario description	<ol style="list-style-type: none"> 1. The EV Driver is authorized by the Charging Station and/or CSMS. 2. The lock Connector attempt fails. 3. A NotifyEventRequest for the ConnectorPlugRetentionLock component, variable = Problem, value = true.
5	Prerequisite(s)	<p>Charging Cable plugged in (status = Occupied) Charging Station has the ConnectorPlugRetentionLock component defined in its Device Model. MonitoringLevel is set to a level that a connector lock event failure will be reported.</p>
6	Postcondition(s)	Transaction is not started and connector lock event failure is reported.

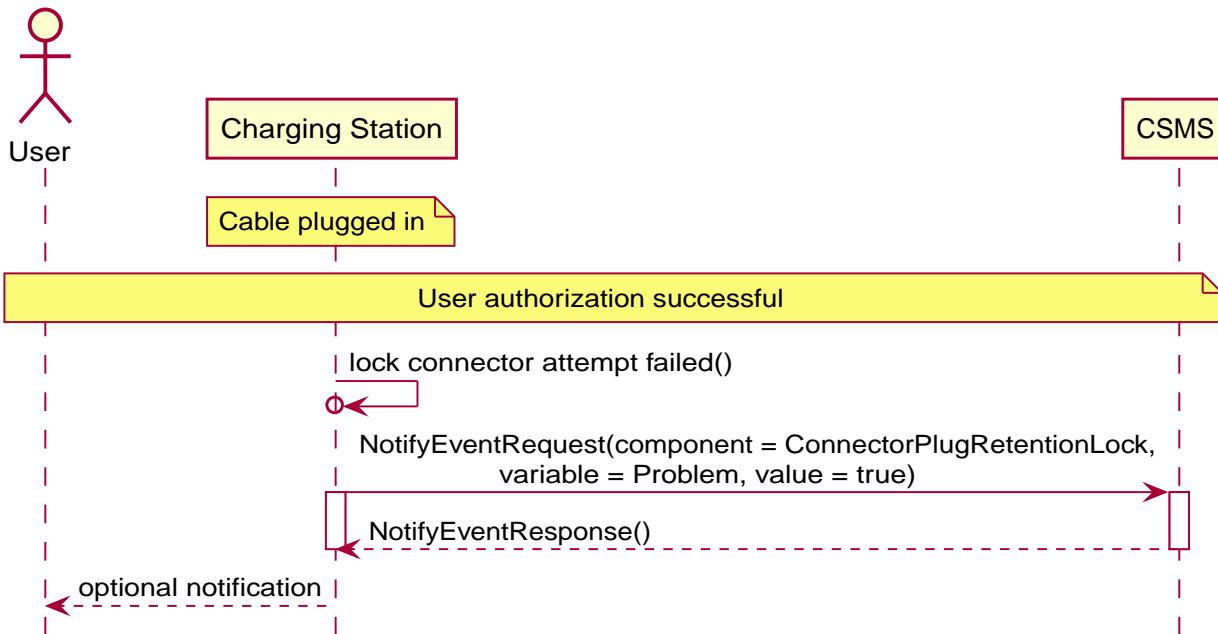


Figure 78. Sequence Diagram: Lock Failure

7	Error handling	n/a
8	Remark(s)	It is advisable to provide some sort of notification to the EV Driver ("cable cannot be locked").

G05 - Lock Failure - Requirements

Table 145. G05 - Requirements

ID	Precondition	Requirement definition	Note
G05.FR.01	If the locking of the connector retention lock fails.	The Charging Station SHALL NOT start charging.	
G05.FR.02	G05.FR.01	The Charging Station SHALL send a NotifyEventRequest to the CSMS for the ConnectorPlugRetentionLock component with variable = Problem, Value = True.	
G05.FR.03	G05.FR.02	The CSMS SHALL respond with a NotifyEventResponse .	

ID	Precondition	Requirement definition	Note
G05.FR.04	G05.FR.01	The Charging Station MAY show an optional notification to the EV Driver.	To notify the EV driver of the lock failure.

H. Reservation

1. Introduction

This Functional Block describes the reservation functionality of OCPP. The reservation functionality enables an EV Driver to make a reservation of a Charging Station/EVSE, ensuring an available Connector at a Charging Station when he arrives.

With Charging Stations not being abundantly available, and EVs having limited range, EV Drivers plan their trips from Charging Station to Charging Station. They need to know for sure they can use a Charging Station they plan to go to. They don't like it when another EV Driver has started using the Charging Station in the time they were traveling to the Charging Station.

For the EV Driver it is useful to be able to reserve a specific Type of Connector, or, when the EV Driver has no preference, an unspecified EVSE at a Charging Station. So he knows for sure he can charge at the Charging Station when he arrives.

2. Use cases & Requirements

H01 - Reservation

Table 146. H01 - Reservation

No.	Type	Description
1	Name	Reservation
2	ID	H01
	Functional block	H. Reservation
3	Objective(s)	To ensure the EV Driver can charge his EV at a Charging Station, the EV Driver can make a reservation until a certain expiry time.
4	Description	This use case describes how a Charging Station can be reserved for a specific IdTokenType .
5	Actors	Charging Station, CSMS, EV Driver
S1	Scenario objective	Reserve an unspecified EVSE at a Charging Station
	Scenario description	<ol style="list-style-type: none"> EV Driver asks the CSMS to reserve an unspecified EVSE at the Charging Station. The CSMS sends ReserveNowRequest without <code>evseld</code> to a Charging Station. Upon receipt of ReserveNowRequest, the Charging Station responds with ReserveNowResponse with status Accepted.
	Prerequisite(s)	The Charging Station has at least one available EVSE
	Postcondition(s)	<p>Successful postcondition: The Charging Station has accepted the ReserveNowRequest</p> <p>Failure postcondition: The Charging Station has rejected the ReserveNowRequest</p>

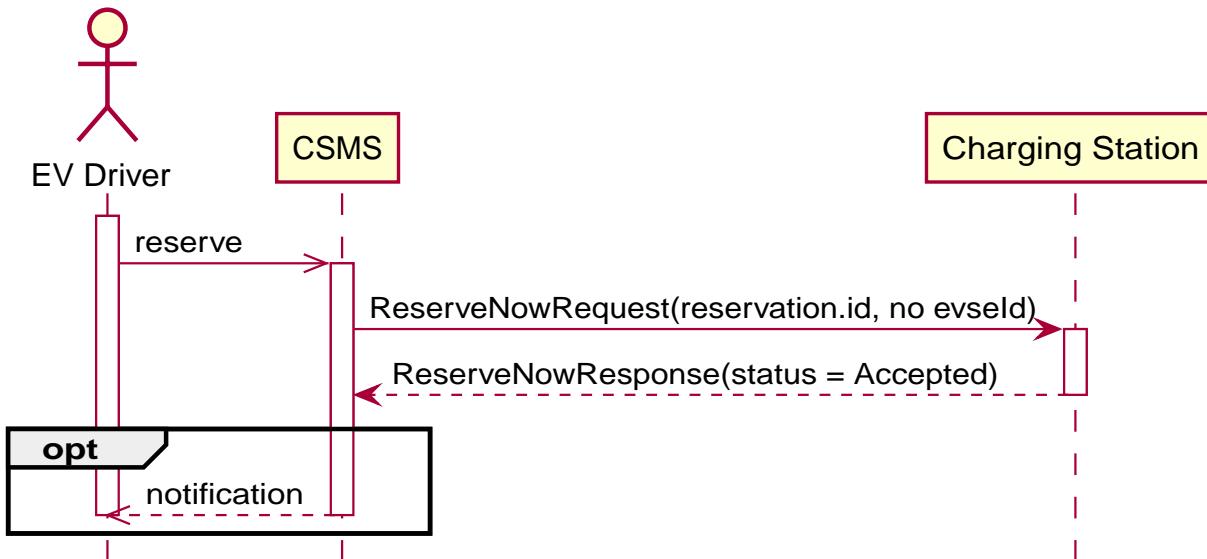


Figure 79. Sequence Diagram: S1 - Reserve a unspecified EVSE at a Charging Station

S2	Scenario objective	Reserve a specific EVSE at a Charging Station
	Scenario description	<ol style="list-style-type: none"> EV Driver asks the CSMS to reserve a specific EVSE at the Charging Station. The CSMS sends ReserveNowRequest with a EVSE to a Charging Station. Upon receipt of ReserveNowRequest, the Charging Station responds with ReserveNowResponse with status Accepted. The Charging Station sends StatusNotificationRequest with the status Reserved for all Connectors of that EVSE. The CSMS responds with StatusNotificationResponse to the Charging Station.
	Prerequisite(s)	The specified EVSE of the Charging Station has status Available

	<p>Postcondition(s)</p> <p>Successful postcondition: The Charging Station has accepted the ReserveNowRequest AND sent StatusNotificationRequests with status <i>Reserved</i>.</p> <p>Failure postcondition: The Charging Station has rejected the ReserveNowRequest OR The Charging Station has NOT sent StatusNotificationRequests with status <i>Reserved</i>.</p>
--	---

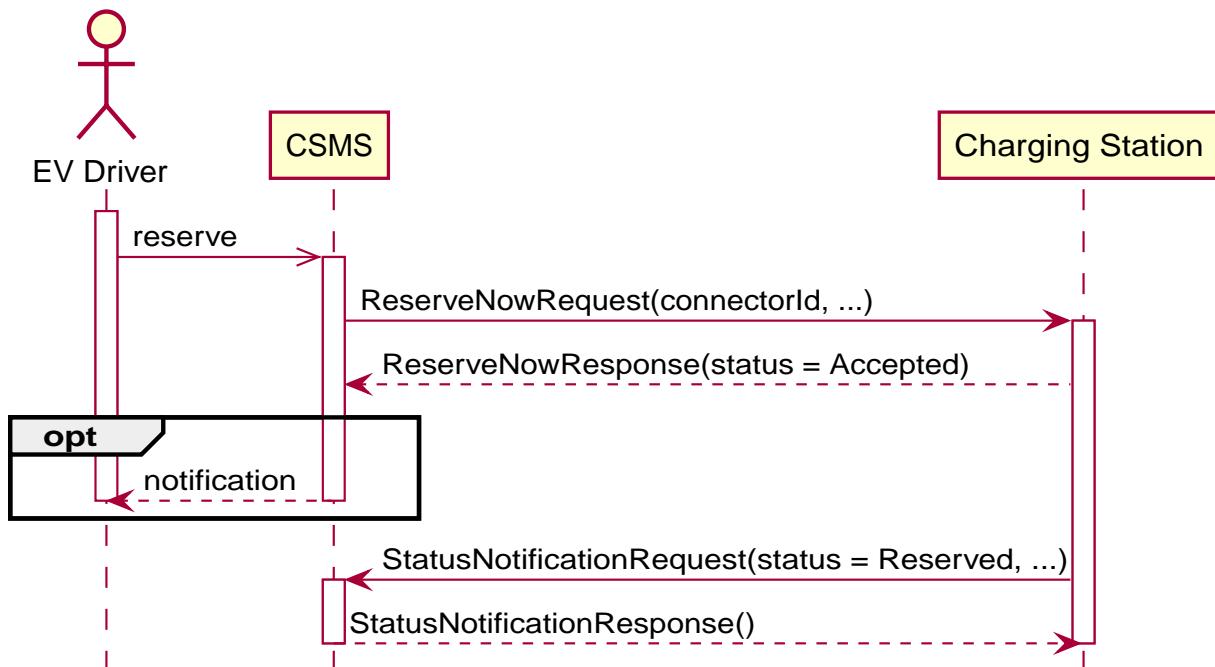


Figure 80. Sequence Diagram: S2 - Reserve a specified EVSE at a Charging Station

S3	Scenario objective	Reserve a connector type at a Charging Station
	Scenario description	1. EV Driver asks the CSMS to reserve a connector type at the Charging Station. 2. The CSMS sends ReserveNowRequest with a connector type to a Charging Station. 3. Upon receipt of ReserveNowRequest , the Charging Station responds with ReserveNowResponse with status Accepted.
	Prerequisite(s)	The Charging Station has at least one available EVSE with the specified connector type
	Postcondition(s)	<p>Successful postcondition: The Charging Station has accepted the ReserveNowRequest</p> <p>Failure postcondition: The Charging Station has rejected the ReserveNowRequest</p>



Figure 81. Sequence Diagram: S3 - Reserve a connector type at a Charging Station

6	Error handling	
7	Remark(s)	It is RECOMMENDED to validate the Identifier with an AuthorizeRequest after reception of ReserveNowRequest and before the start of the transaction.

H01 - Reservation - Requirements

Table 147. H01 - Requirements

ID	Precondition	Requirement definition	Note
H01.FR.01	If the Charging Station is configured not to accept reservations.	The Charging Station SHALL return <i>Rejected</i> .	
H01.FR.02	If the <i>id</i> in the ReserveNowRequest matches a reservation in the Charging Station.	The Charging Station SHALL replace that reservation with the new reservation in the request.	
H01.FR.03	If the <i>id</i> in the ReserveNowRequest does not match any reservation in the Charging Station.	The Charging Station SHALL return the status value <i>Accepted</i> if it succeeds in reserving an EVSE.	
H01.FR.04	If the Charging Station receives a ReserveNowRequest without <i>evseld</i> AND at least one EVSE is <i>Available</i> AND H01.FR.18	The Charging Station SHALL accept the reservation AND respond with a ReserveNowResponse with status <i>Accepted</i> .	
H01.FR.06	If the Charging Station receives a ReserveNowRequest with a connector type AND at least one EVSE with the specified connector type is <i>Available</i> AND H01.FR.18	The Charging Station SHALL accept the reservation AND respond with a ReserveNowResponse with status <i>Accepted</i> .	
H01.FR.07	When the Charging Station has Accepted a ReserveNowRequest without <i>evseld</i>	The Charging Station SHALL make sure that at any time during the validity of the reservation, one EVSE remains available for the reserved IdTokenType .	
H01.FR.09	When the Charging Station has Accepted a ReserveNowRequest with a connector type	The Charging Station SHALL make sure that at any time during the validity of the reservation, one Connector with the specified type remains available for the reserved IdTokenType .	
H01.FR.11	When receiving a ReserveNowRequest AND (all) targeted EVSEs have status <i>Reserved</i> or <i>Occupied</i>	The Charging Station SHALL return <i>Occupied</i> .	
H01.FR.12	When receiving a ReserveNowRequest AND (all) targeted EVSEs have status <i>Faulted</i>	The Charging Station SHALL return <i>Faulted</i> .	

ID	Precondition	Requirement definition	Note
H01.FR.14	When receiving a ReserveNowRequest AND (all) targeted EVSEs have status <i>Unavailable</i>	The Charging Station SHALL return <i>Unavailable</i> .	
H01.FR.15	If a transaction for the reserved IdTokenType is started.	The Charging Station SHALL send the reservationId in a TransactionEventRequest .	To notify the CSMS that the reservation is terminated. See E. Transactions .
H01.FR.16	When the status of a targeted EVSE changes to <i>Faulted</i>	The Charging Stations SHALL cancel the reservation AND send a ReservationStatusUpdate with status <i>Removed</i> .	
H01.FR.17	When the status of a targeted EVSE changes to <i>Unavailable</i>	The Charging Stations SHALL cancel the reservation AND send a ReservationStatusUpdate with status <i>Removed</i> .	
H01.FR.18	If the Configuration Variable: ReservationNonEvseSpecific is set to <i>true</i> .	The Charging Station SHALL accept reservations on an unspecified EVSE.	
H01.FR.19	If the Configuration Variable: ReservationNonEvseSpecific is not set or set to <i>false</i> .	The Charging Station SHALL reject reservations on an unspecified EVSE.	
H01.FR.20	H01.FR.04 AND amount of EVSEs available equals the amount of reservations	The Charging Station SHALL send a StatusNotificationRequest with <i>connectorStatus</i> = Reserved for all connectors of the EVSE.	If an EVSE is reserved, all of its connectors are reported as reserved.
H01.FR.23	If the Charging Station receives a ReserveNowRequest for <i>evsel</i> AND this EVSE is <i>Available</i>	The Charging Station SHALL respond with a ReserveNowResponse with status <i>Accepted</i> AND SHALL send a StatusNotificationRequest with <i>connectorStatus</i> = Reserved for all connectors of the EVSE.	If an EVSE is reserved, all of its connectors are reported as reserved.
H01.FR.24	H01.FR.06 AND amount of reservations for a specific <i>connectorType</i> equals the amount of available EVSEs with that specific <i>connectorType</i>	The Charging Station SHALL send a StatusNotificationRequest with <i>connectorStatus</i> = Reserved for all connectors of the EVSEs with the specific <i>connectorType</i> .	If an EVSE is reserved for a specific <i>connectorType</i> , all connectors on the EVSE are reported as reserved.

H02 - Cancel Reservation

Table 148. H02 - Cancel Reservation

No.	Type	Description
1	Name	Cancel Reservation
2	ID	H02
	Functional block	H. Reservation
3	Objective(s)	To cancel a reservation on a Charging Station.
4	Description	This use case describes how an EV Driver can cancel an existing reservation. The CSMS can cancel the reservation the EV Driver has on a Charging Station.
	Actors	Charging Station, CSMS, EV Driver
	Scenario description	<ol style="list-style-type: none"> EV Driver asks the CSMS to cancel a reservation. To cancel a reservation the CSMS sends CancelReservationRequest to the Charging Station. If the Charging Station has a reservation matching the reservationId in the request PDU, it returns the status Accepted. If a specific EVSE was reserved for this reservation, the Charging Station sends StatusNotificationRequest with the status Available for all the Connectors of that EVSE. The CSMS responds with StatusNotificationResponse to the Charging Station. The reservation is cancelled.
5	Prerequisite(s)	<ul style="list-style-type: none"> The Functional Block <i>Reservation</i> is installed. EV Driver has a reservation at the Charging Station.
6	Postcondition(s)	<p>Successful postcondition: The CSMS was able to cancel the EV Driver's reservation at the Charging Stations.</p> <p>Failure postcondition: n/a.</p>

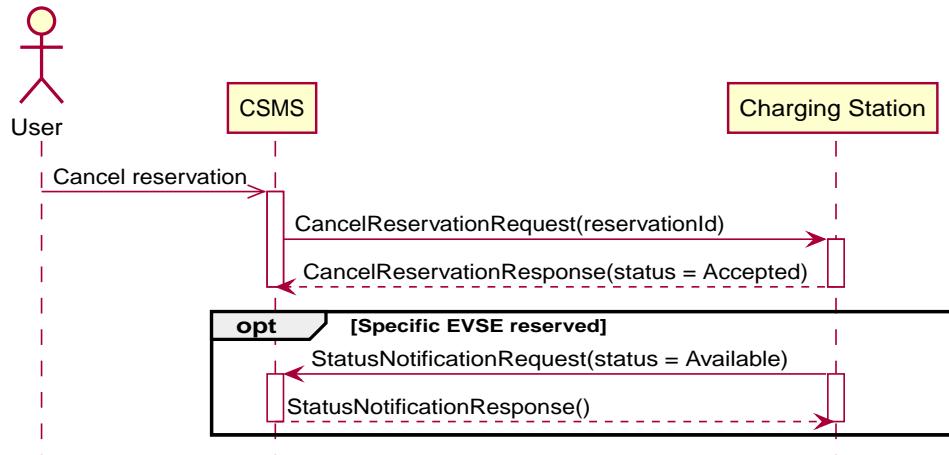


Figure 82. Sequence Diagram: Cancel Reservation

7	Error handling	n/a
8	Remark(s)	The Charging Station does not send a ReservationStatusUpdate , because it was explicitly cancelled by CSMS, so it is already aware of the event.

H02 - Cancel Reservation - Requirements

Table 149. H02 - Requirements

ID	Precondition	Requirement definition
H02.FR.01	The Charging Station has received a CancelReservationRequest and no matching reservationId.	The Charging Station SHALL return <i>Rejected</i> .

ID	Precondition	Requirement definition
H02.FR.02	If a Charging Station receives a CancelReservationRequest with a valid, known reservationId.	The reservation SHALL be cancelled.

H03 - Use a reserved EVSE

No.	Type	Description
1	Name	Use a reserved EVSE
2	ID	H03
	<i>Functional block</i>	H. Reservation
3	Objective(s)	Use a reserved EVSE
4	Description	This use cases covers how a reserved EVSE can be used based on IdToken and GroupIdToken information.
	Actors	Charging Station, CSMS, EV Driver
S1	Scenario objective	Use an EVSE reserved by the same IdToken
	<i>Scenario description</i>	<p>1. The CSMS sends a ReserveNowRequest to a Charging Station to reserve an EVSE for use by a specific IdTokenType.</p> <p>2. Upon receipt of the ReserveNowRequest, the Charging Station responds with a ReserveNowResponse.</p> <p>3. When a specific EVSE is reserved for this reservation, the Charging Station sends a StatusNotificationRequest with the status <i>Reserved</i> for all the Connectors of that EVSE.</p> <p>4. The CSMS responds with a StatusNotificationResponse to the Charging Station.</p> <p>5. The EV Driver presents an IdTokenType at the Charging Station, and the IdTokenType is the same as the reservation's IdTokenType, the Charging Station recognizes the IdTokenType and starts charging and E02 - Start Transaction - Cable Plugin First applies.</p>
5	Prerequisite(s)	n/a
6	Postcondition(s)	n/a

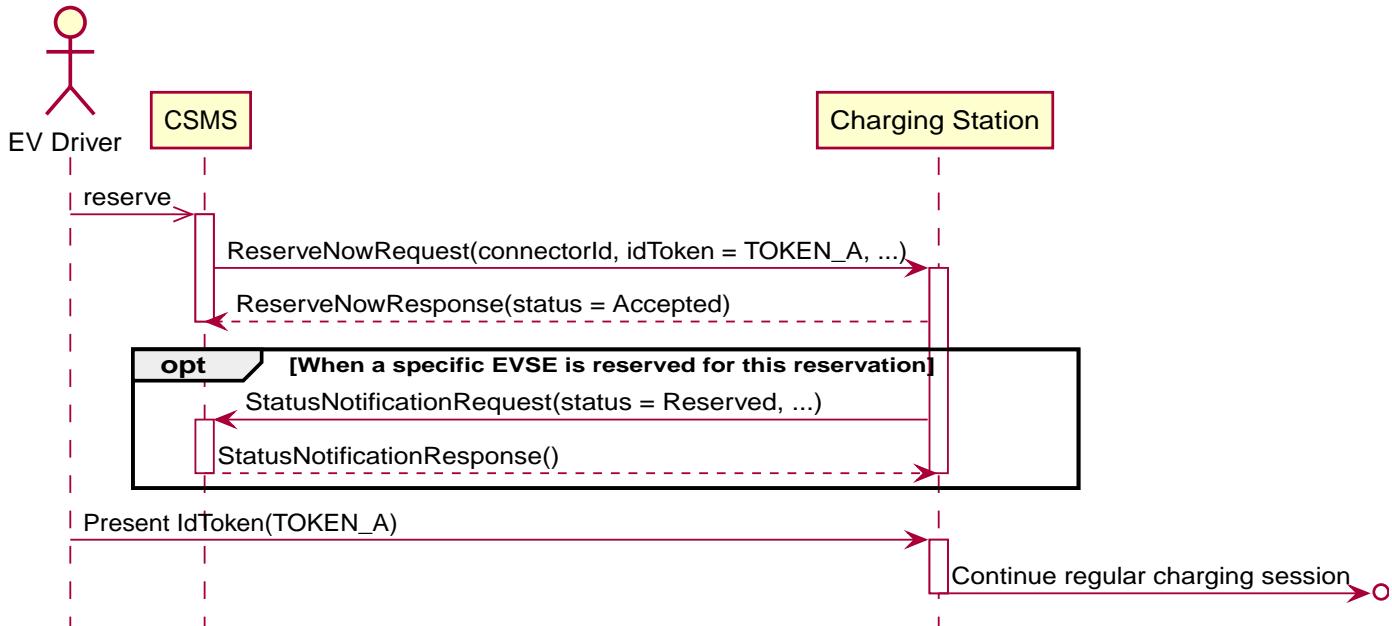


Figure 83. Sequence Diagram: Use a reserved EVSE with IdToken

S2	Scenario objective	Use an EVSE reserved by the same GroupIdToken
----	--------------------	---

	Scenario description	<ol style="list-style-type: none"> 1. The CSMS sends a ReserveNowRequest with the GroupId to a Charging Station to reserve a EVSE for use by a specific IdTokenType. 2. Upon receipt of the ReserveNowRequest, the Charging Station responds with a ReserveNowResponse. 3. When a specific EVSE is reserved for this reservation, the Charging Station sends a StatusNotificationRequest with the status Reserved for all the Connectors of that EVSE. 4. The CSMS responds with a StatusNotificationResponse to the Charging Station. 5. The EV Driver presents an IdTokenType at the Charging Station, and the IdTokenType is different from the reservation's IdTokenType, the Charging Station sends an AuthorizeRequest to the CSMS. 6. The CSMS responds with an AuthorizeResponse. This response message includes the GroupId. 7. Based on the matching GroupId information in both responses, the Charging Station starts charging and E02 - Start Transaction - Cable Plugin First applies.
5	Prerequisite(s)	n/a
6	Postcondition(s)	n/a

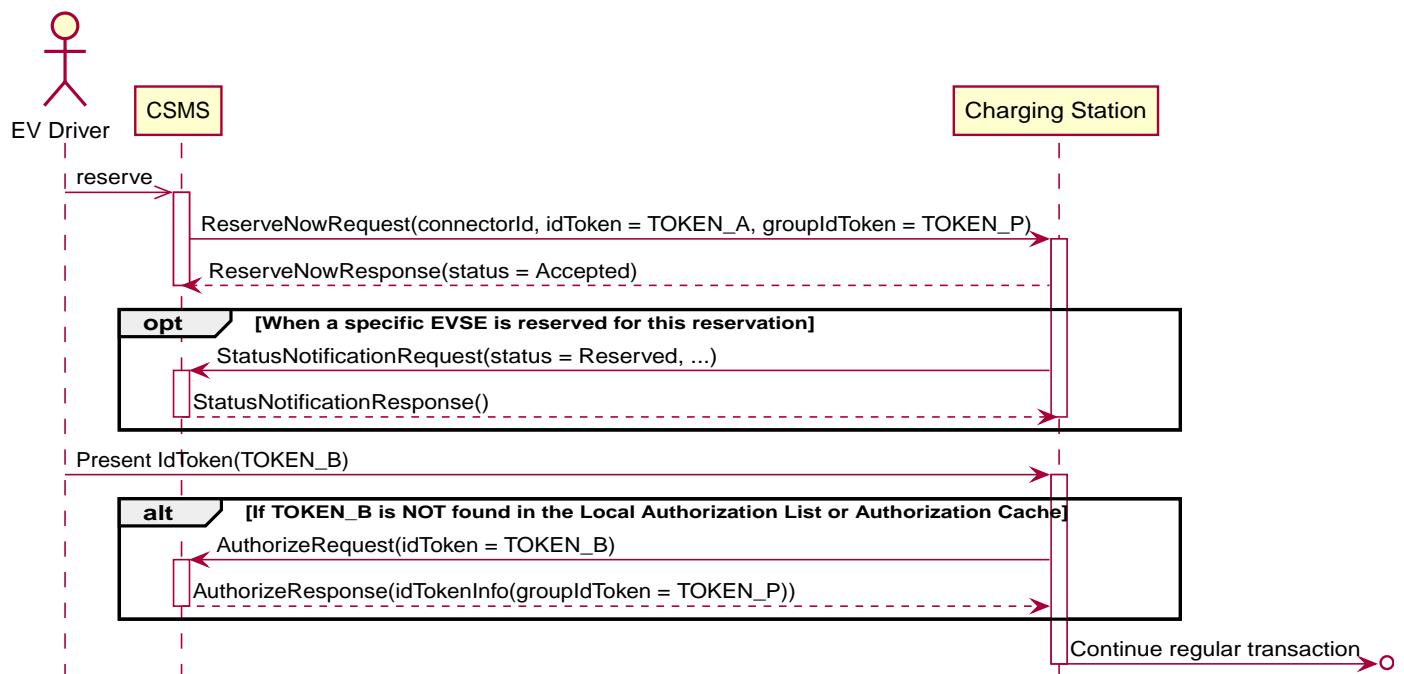


Figure 84. Sequence Diagram: Use a reserved EVSE with GroupId

7	Error handling	n/a
8	Remark(s)	n/a

H03 - Use a reserved EVSE - Requirements

Table 150. H03 - Requirements

ID	Precondition	Requirement definition
H03.FR.01	Reservation is pending for a specific <i>idToken</i> for a specific <i>evseld</i>	The Charging Station SHALL allow charging on that EVSE when <i>IdToken</i> presented for authorization matches the specific <i>idToken</i> from the reservation.
H03.FR.02	Reservation is pending for a specific <i>idToken</i> for a specific <i>connectorType</i>	The Charging Station SHALL allow charging on an EVSE with a connector of type <i>connectorType</i> when <i>IdToken</i> presented for authorization matches the specific <i>idToken</i> from the reservation.
H03.FR.03	Reservation is pending for a specific <i>idToken</i> without a specific <i>evseld</i> or <i>connectorType</i>	The Charging Station SHALL allow charging on an EVSE when <i>IdToken</i> presented for authorization matches the specific <i>idToken</i> from the reservation.
H03.FR.04	H03.FR.01 AND attribute <i>groupIdToken</i> in reservation has a value	The Charging Station SHALL allow charging on that EVSE when <i>IdToken</i> presented for authorization matches the specific <i>idToken</i> from the reservation or when the associated <i>groupIdToken</i> matches.
H03.FR.05	H03.FR.02 AND attribute <i>groupIdToken</i> in reservation has a value	The Charging Station SHALL allow charging on an EVSE with a connector of type <i>connectorType</i> when <i>IdToken</i> presented for authorization matches the specific <i>idToken</i> from the reservation or when the associated <i>groupIdToken</i> matches.
H03.FR.06	H03.FR.03 AND attribute <i>groupIdToken</i> in reservation has a value	The Charging Station SHALL allow charging on any EVSE when <i>IdToken</i> presented for authorization matches the specific <i>idToken</i> from the reservation or when the associated <i>groupIdToken</i> matches.
H03.FR.07	If attribute <i>groupIdToken</i> in the reservation has a value (it is optional).	In order to determine the <i>groupIdToken</i> that is associated with an incoming <i>IdToken</i> , the Charging Station MAY look it up in its Local Authorization List or Authorization Cache.
H03.FR.08	H03.FR.07 AND If it is not found in the Local Authorization List or Authorization Cache.	The Charging Station SHALL send an AuthorizeRequest for the incoming <i>IdToken</i> to the CSMS in order to get its associated <i>groupIdToken</i> .

H04 - Reservation Ended, not used

No.	Type	Description
1	Name	Reservation Ended, not used
2	ID	H04
	Functional block	H. Reservation
3	Objective(s)	To enable a Charging Station to notify the CSMS about a reservation that has expired.
4	Description	This use cases covers how the Charging Station notifies the CSMS about a reservation, that has ended/timed out before the EV Driver starts using the Charging Station.
	Actors	Charging Station, CSMS
	Scenario description	<ol style="list-style-type: none"> 1. The Charging Station has a reservation. 2. The expiryDate of the reservation is reached. 3. The Charging Station removes the reservation . 4. If a specific EVSE was reserved for this reservation, the Charging Station makes the EVSE available again and notifies the CSMS about this by sending a StatusNotificationRequest with the status <i>Available</i> for that all the Connectors of that EVSE. 5. The CSMS responds with a StatusNotificationResponse. 6. The Charging Station sends a ReservationStatusUpdateRequest with status <i>Expired</i> to the CSMS. 7. The CSMS responds with a ReservationStatusUpdateResponse.
5	Prerequisite(s)	n/a
6	Postcondition(s)	n/a

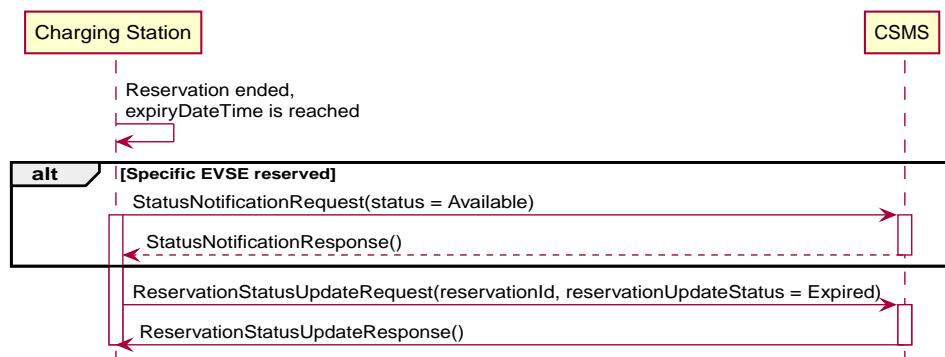


Figure 85. Sequence Diagram: Reservation Ended, not used

7	Error handling	n/a
8	Remark(s)	n/a

H04 - Reservation Ended, not used - Requirements

Table 151. H04 - Requirements

ID	Precondition	Requirement definition
H04.FR.01	The reservation ends (expiryDateTime reached)	The Charging Station SHALL send a ReservationStatusUpdateRequest with status <i>Expired</i> .
H04.FR.02	H04.FR.01 AND If a specific EVSE was reserved for this reservation	The Charging Station SHALL allow charging again on this EVSE.
H04.FR.03	H04.FR.02	The Charging Station SHALL send a StatusNotificationRequest with status <i>Available</i> to the CSMS, notifying the CSMS the all the connectors of this EVSE are available again for any EV Driver.

I. TariffAndCost

1. Introduction

This Functional Block provides tariff and cost information to an EV Driver, when a Charging Station is capable of showing this on a display.

Before a driver starts charging he needs to be given tariff information, given detailed prices for all the components that make up the tariff plan applicable to this driver at this Charging Station. As this is a human readable text message, it can also be used for other things, like a personal welcome message.

Some business cases might require the EV Driver to be shown the running total cost during charging, updated at a regular, fitting interval. When the EV Driver stops charging, he needs to be shown to the total cost of the just stopped transaction.

All tariffs and costs are in the currency configured in the Configuration Variable [Currency](#).

1.1. Why no structured tariff information?

Because tariff structures can become very complex it will be difficult to convert these to human-readable text in the Charging Station. The CSO is the owner of the tariffs and should be able to provide the Charging Station with a human-readable tariff text. If the CSO is not able to generate human-readable texts from its own tariffs, how can a Charging Station be expected to be able to this. That is why we have kept the complexity of tariffs out of OCPP.

2. Use cases & Requirements

I01 - Show EV Driver-specific Tariff Information

No.	Type	Description
1	Name	Show EV Driver-specific Tariff Information
2	ID	I01
	Functional block	I. Tariff and Cost
3	Objective(s)	To show an EV Driver-specific tariff before the start of a transaction.
4	Description	When an EV Driver wants to charge an EV he wants to know how much charging will cost him at the Charging Station he is at. The EV Driver is authenticated by his (RFID) token. The Charging Station asks the CSMS for information about the presented token. The CSMS returns information about the token, including the tariff applicable to this EV Driver.
	Actors	Charging Station, CSMS, EV Driver
	Scenario description	<ol style="list-style-type: none"> 1. The EV Driver wants to charge an EV, he presents his <code>IdTokenType</code>. 2. The Charging Station sends <code>AuthorizeRequest</code> to the CSMS to request authorization. 3. Upon receipt of <code>AuthorizeRequest</code>, the CSMS responds with <code>AuthorizeResponse</code>. This response message indicates whether or not the <code>IdTokenType</code> is accepted by the CSMS, and reports the EV Driver-specific tariff in the <code>PersonalMessage</code> field. 4. The Charging Station shows the EV Driver-specific tariff to the EV Driver.
	Alternative scenario(s)	I04 - Show Fallback Tariff Information
5	Prerequisite(s)	The Charging Station supports Tariff Information
6	Postcondition(s)	<p>Successful postcondition: The EV Driver is authorized, knows which tariff is applicable for him/her and can start charging.</p> <p>Failure postcondition: If the authorization status is other than Accepted, the EV Driver can not start and might not know the tariff.</p>

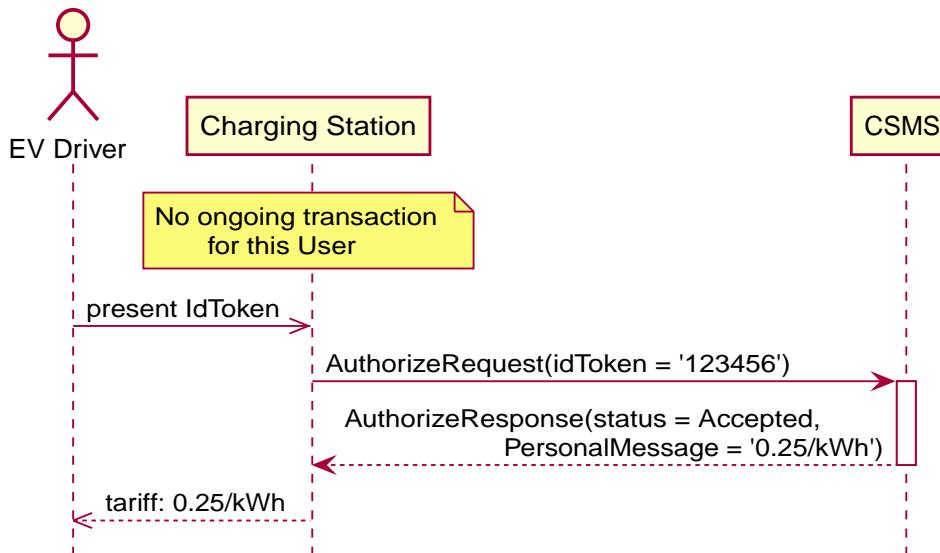


Figure 86. Sequence Diagram: Show EV Driver-specific tariff information

7	Error Handling	n/a
8	Remarks	<p>The tariff information presented this way might be equal to any token presented.</p> <p>If known, and applicable, it is advisable to show the tariff information in a language understood by the EV Driver.</p> <p>It is advisable to give the driver the option to cancel the transaction when he does not agree with the tariff. This could be not plugging in the cable, or a cancel button in the user interface etc. As long as it is clear to the driver how a transaction can be canceled.</p>

I01 - Show EV Driver-specific Tariff Information - Requirements

ID.	Precondition	Requirements
I01.FR.01		The CSMS MAY send EV Driver-specific tariff information in the PersonalMessage field of an AuthorizeResponse message.
I01.FR.02		The CSMS SHALL only send the tariff information if the Charging Station supports the tariff or DisplayMessage functionality.
I01.FR.03	I01.FR.01	The Charging Station SHALL show the EV Driver-specific tariff information to the EV Driver.

I02 - Show EV Driver Running Total Cost During Charging

No.	Type	Description
1	Name	Show EV Driver Running Total Cost During Charging
2	ID	I02
	Functional block	I. Tariff and Cost
3	Objectives	To show an EV Driver the running total cost during charging
4	Description	While a transaction is ongoing, the driver wants to know how much the running total cost is, updated at a relevant interval.
	Actors	Charging Station, CSMS, EV Driver
	Scenario description	<ol style="list-style-type: none"> Every Y seconds the CSMS sends a CostUpdatedRequest to the Charging Station to update the current total cost. Upon receipt of the CostUpdatedRequest, the Charging Station responds with a CostUpdatedResponse. The Charging Station shows the current total cost to the EV Driver.
	Alternative scenario	<ol style="list-style-type: none"> Upon receipt of a TransactionEventRequest with eventType = Updated the CSMS returns the running cost corresponding to the timestamp and meterValue in the field totalCost in the TransactionEventResponse. The Charging Station shows the current total cost to the EV Driver.
5	Prerequisites	The Charging Station supports Tariff Information Ongoing transaction
6	Postcondition(s)	<p>Successful postcondition: The EV Driver knows the running total cost during charging.</p> <p>Failure postcondition: Total cost not known to the EV Driver during charging.</p>

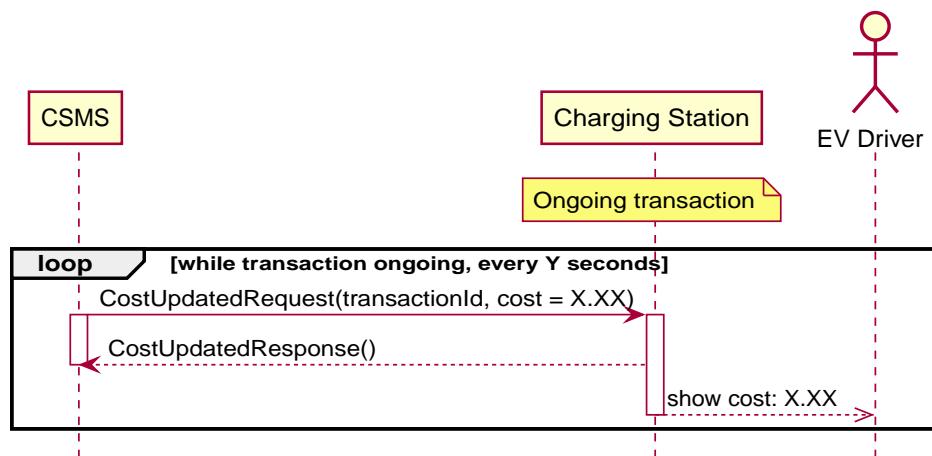


Figure 87. Sequence Diagram: Show EV Driver Running Total Cost During Charging

7	Error Handling	n/a
8	Remarks	Updating the running cost very often will create a lot of messages, which might result in high mobile data cost.

I02 - Show EV Driver Running Total Cost During Charging - Requirements

ID.	Precondition	Requirements
I02.FR.01		The CSMS SHALL send either a CostUpdatedRequest at a relevant interval/moment or return the running cost in a TransactionEventResponse . This might depend on the charging speed, running cost, etc.
I02.FR.02	Upon receipt of a CostUpdatedRequest message.	The Charging Station SHALL respond with a CostUpdatedResponse message.
I02.FR.03	I02.FR.02	The Charging Station SHALL show the current total cost to the EV Driver.
I02.FR.04	When running cost is reported in TransactionEventResponse	The Charging Station SHALL show the current running cost to the EV Driver.

I03 - Show EV Driver Final Total Cost After Charging

No.	Type	Description
1	Name	Show EV Driver Final Total Cost After Charging
2	ID	I03
	Functional block	I. Tariff and Cost
3	Objectives	To show an EV Driver the total cost after the transaction is finished.
4	Description	An EV Driver stops an ongoing transaction by presenting his identification token (for example RFID). The transaction is stopped and the total cost of the transaction is shown to the EV Driver.
	Actors	Charging Station, CSMS, EV Driver
	Scenario description	<ol style="list-style-type: none"> 1. The EV Driver presents an IdTokenType to stop the transaction. 2. The Charging Station sends TransactionEventRequest (eventType = Ended) 3. The CSMS responds with TransactionEventResponse containing the total cost of the transaction. 4. The Charging Station shows the total cost to the EV Driver.
	Alternative scenario's	I05 - Show Fallback Total Cost Message
5	Prerequisites	The Charging Station supports Tariff Information Ongoing transaction
6	Postcondition(s)	<p>Successful postcondition: The EV Driver knows the total cost of the transaction.</p> <p>Failure postcondition: The EV Driver does NOT know the total cost of the transaction.</p>

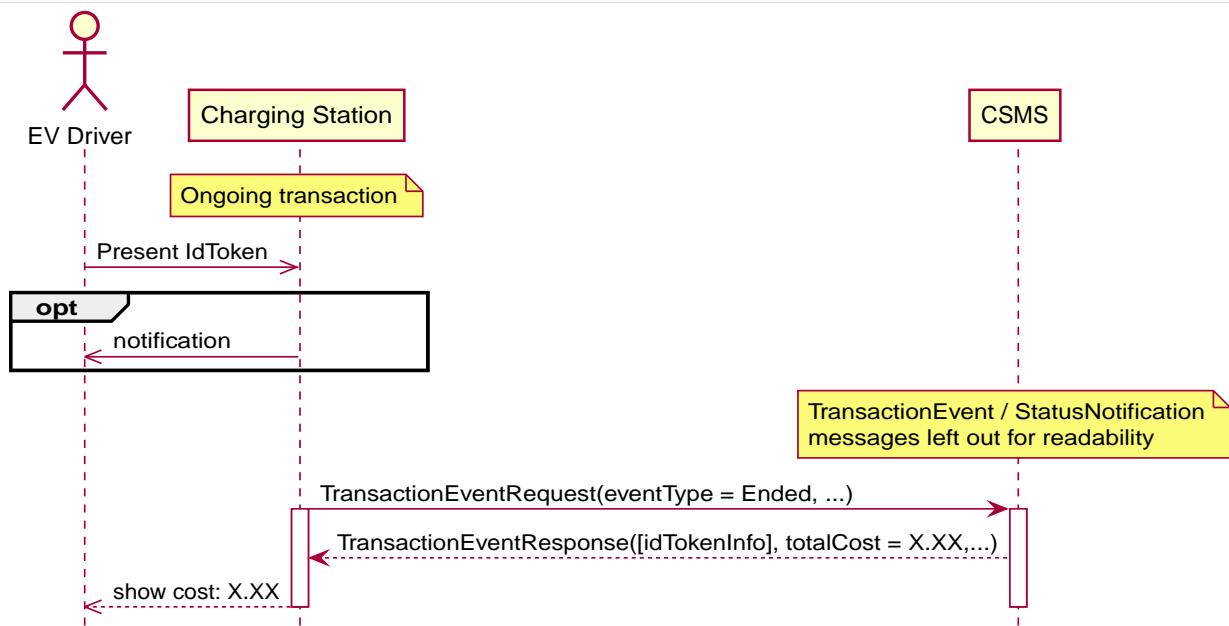


Figure 88. Sequence Diagram: Show EV Driver Final Total Cost After Charging

7	Error Handling	n/a
8	Remarks	<p>If the Charging Station was offline when the transaction ended and the TransactionEventResponse with <code>totalCost</code> is received when the Charging Station comes back online some time after that, then there is no use in displaying the cost, because the user has likely left already. A similar situation applies when <code>TxStopPoint</code> is defined as ParkingBayOccupancy, in which case the EV must leave the Charging Station to cause the transaction to end.</p> <p>The scenario description and sequence diagram above are based on the Configuration Variable for stop transaction being configured as follows. TxStopPoint: ParkingBayOccupancy, EVConnected, Authorized This use-case is also valid for other configurations, but then the transaction might stop at another moment, which might change the sequence in which messages are sent. For more details see the use case: E06 - Stop Transaction options</p>

I03 - Show EV Driver Final Total Cost After Charging - Requirements

ID.	Precondition	Requirements
I03.FR.01	When transaction is stopped	The Charging Station SHALL send a TransactionEventRequest (<code>eventType = Ended</code>) to the CSMS.
I03.FR.02	I03.FR.01 AND When Total Cost is known to the CSMS.	The CSMS SHALL send the total cost of the transaction in the <code>totalCost</code> field of the TransactionEventResponse message.
I03.FR.03	I03.FR.02 AND Charging Station was online when transaction stopped	The Charging Station SHALL display the total cost to the EV Driver.
I03.FR.04		To indicate a free transaction, the CSMS SHALL set <code>totalCost</code> to 0.00. Thus omitting <code>totalCost</code> does not imply that the transaction was free.
I03.FR.05	I02.FR.02 AND <code>TxStopPoint</code> is defined as ParkingBayOccupancy	The Charging Station SHOULD NOT display the total cost to the EV Driver. (Driver has left already).

I04 - Show Fallback Tariff Information

No.	Type	Description
1	Name	Show Fallback Tariff Information
2	ID	I04
	Functional block	I. Tariff and Cost

No.	Type	Description
3	Objective(s)	To show an EV Driver some information, generic tariff, a message etc., when the Charging Station cannot retrieve tariff information for this EV Driver.
4	Description	When an EV Driver wants to charge an EV, he wants an indication of how much charging will cost him at the Charging Station he is at, but the Charging Station cannot get a specific tariff for this EV Driver (for example: the Charging Station is Offline, or no EV Driver-specific tariff is available). For such scenarios, a fallback tariff information message can be configured in the Charging Station.
	Actors	Charging Station, EV Driver
	Scenario description	<ol style="list-style-type: none"> 1. The EV Driver wants to charge an EV, he presents his IdTokenType. 2. The Charging Station authorizes the EV Driver against the Authorization Cache 3. The Charging Station shows the TariffFallbackMessage to the EV Driver.
	Alternative scenario's	I01 - Show EV Driver-specific Tariff Information
5	Prerequisites	The Charging Station supports Tariff Information the Configuration Variable: TariffFallbackMessage is configured.
6	Postcondition(s)	<p>Successful postcondition: EV Driver has been shown the fallback tariff information message</p> <p>Failure postcondition: EV Driver has no information about the tariff at this Charging Station.</p>

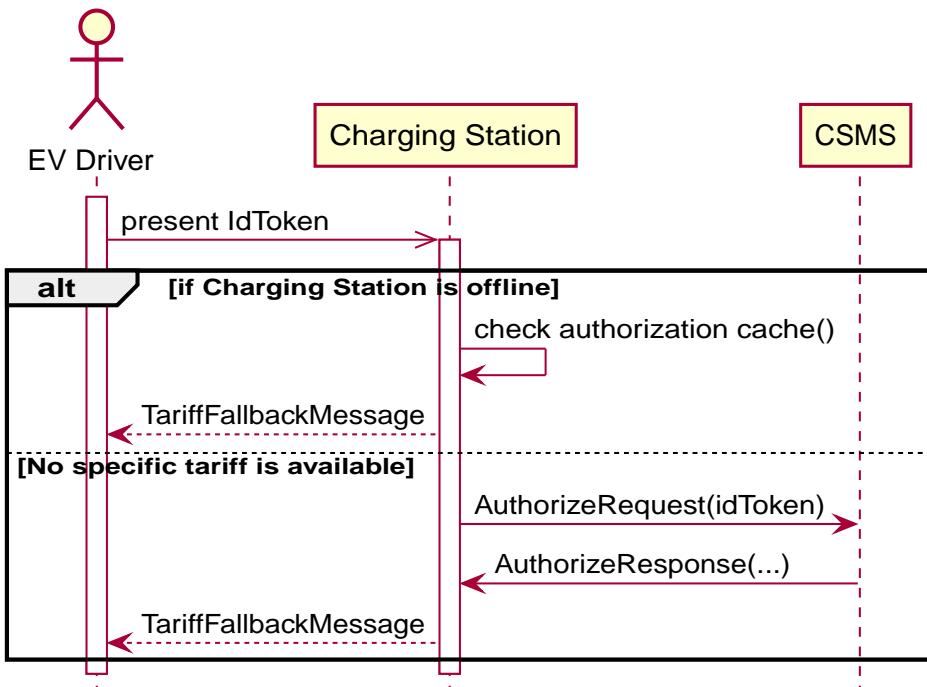


Figure 89. Sequence Diagram: Show Fallback Tariff Information

7	Error Handling	n/a
8	Remarks	n/a

I04 - Show Fallback Tariff Information - Requirements

ID.	Precondition	Requirements
I04.FR.01	When the Charging Station cannot get a specific tariff for the EV Driver (for example: the Charging Station is Offline, or no EV Driver-specific tariff is available.)	The Charging Station SHALL display a fallback tariff information message to the EV Driver, which is configured in the Configuration Variable: TariffFallbackMessage .
I04.FR.02		The CSMS MAY configure the TariffFallbackMessage via the Configuration Variable: TariffFallbackMessage .

I05 - Show Fallback Total Cost Message

No.	Type	Description
1	Name	Show Fallback Total Cost Message
2	ID	I05
	Functional block	I. Tariff and Cost
3	Objectives	To show an EV Driver a message instead of the actual total cost when the Charging Station is <i>Offline</i> when a transaction is stopped.
4	Description	When an EV Driver wants to stop an ongoing transaction, but the Charging Station is <i>Offline</i> . The transaction will be stopped as described earlier. The Charging Station cannot retrieve the total cost for the stopped transaction. The EV Driver needs to be given some message, this message can be configured in the Configuration Variable: TotalCostFallbackMessage .
	Actors	Charging Station, EV Driver
	Scenario description	<ol style="list-style-type: none"> 1. The EV Driver presents IdTokenType to stop the transaction. 2. The Charging Station stops the energy offer. 3. The Charging Station shows the TotalCostFallbackMessage to the EV Driver.
	Alternative scenario's	I03 - Show EV Driver Final Total Cost After Charging
5	Prerequisites	The Charging Station supports Tariff Information The Charging Station is <i>Offline</i> the Configuration Variable: TotalCostFallbackMessage is configured.
6	Postcondition(s)	<p>Successful postcondition: The EV Driver has received a pre-configured fallback message.</p> <p>Failure postcondition: The EV Driver has not received a pre-configured fallback message.</p>

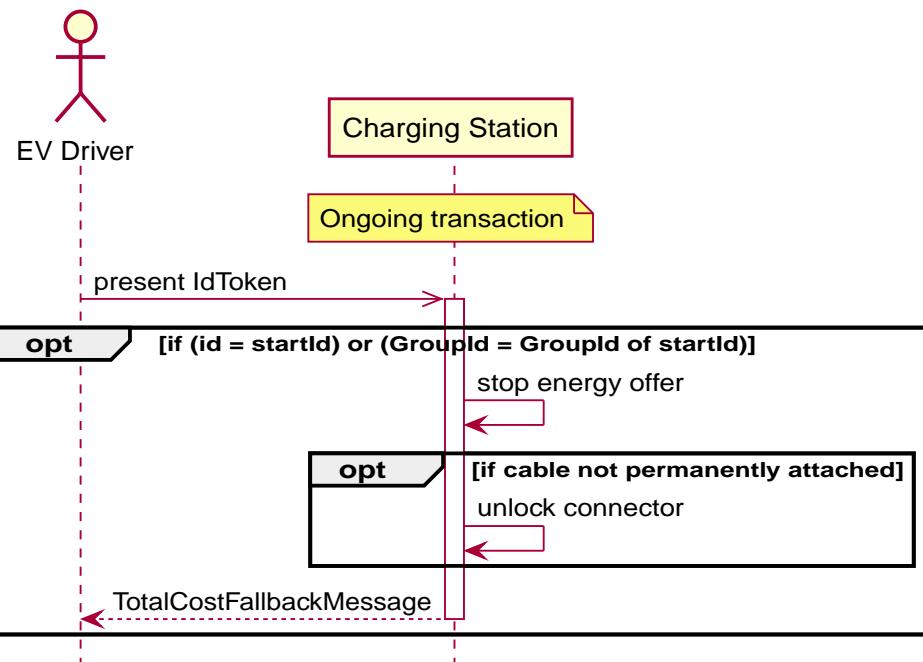


Figure 90. Sequence Diagram: Show Fallback Total Cost Message

7	Error Handling	n/a
8	Remarks	n/a

I05 - Show Fallback Total Cost Message - Requirements

ID.	Precondition	Requirements
I05.FR.01		The CSMS MAY configure the fallback total cost information message via the Configuration Variable: TotalCostFallbackMessage .

ID.	Precondition	Requirements
I05.FR.02	When the Charging Station cannot retrieve the total cost for the stopped transaction, because the Charging Station is offline.	The Charging Station SHALL show a fallback total cost information message to the EV Driver.

I06 - Update Tariff Information During Transaction

No.	Type	Description
1	Name	Update Tariff Information During Transaction
2	ID	I06
	Functional block	I. Tariff and Cost
3	Objectives	To show an EV Driver updated tariff information during a transaction.
4	Description	<p>During charging (especially DC fast charging) it might be useful to show the EV driver updated tariff information when it becomes available.</p> <p>Example: If a tariff has a bandwidth: <i>charging will cost between 0,25 and 0,40 euro/kWh depending on current energy price. Current price is 0,28 euro/kWh.</i></p> <p>Then when the price changing, this tariff information needs to be updated: <i>charging will cost between 0,25 and 0,40 euro/kWh depending on current energy price. Current price is 0,32 euro/kWh.</i></p>
	Scenario description	<ol style="list-style-type: none"> 1. The Charging Station sends <code>TransactionEventRequest (eventType = Updated)</code> messages during the transaction. 2. When the CSMS receives a <code>TransactionEventRequest</code> message it checks if there is updated tariff information available. 3. The CSMS acknowledges with a <code>TransactionEventResponse</code> message, which contains the updated tariff information if available.
5	Prerequisites	<p>The Charging Station supports Tariff Information</p> <p>There is a transaction ongoing</p>
6	Postcondition(s)	<p>Successful postcondition: The updated tariff information is shown to the EV Driver.</p> <p>Failure postcondition: The EV Driver has not been shown the updated tariff information.</p>

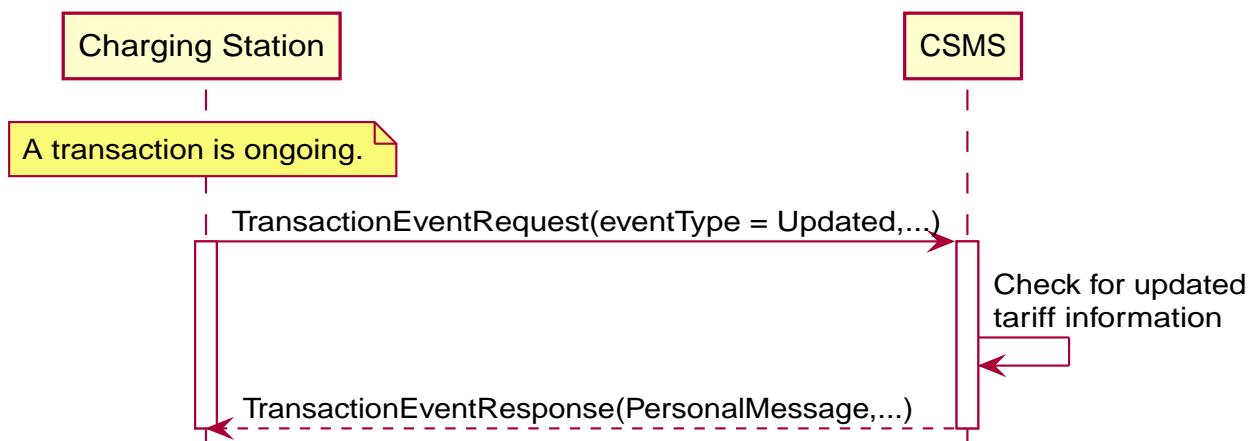


Figure 91. Sequence Diagram: Update Tariff Information During Transaction

7	Error Handling	n/a
8	Remarks	There may be a policy or a legal requirement in place, that the tariff communicated at the start of the transaction must be used for the entire transaction, in which case no updated tariff information should be sent during the transaction.

I06 - Update Tariff Information During Transaction - Requirements

ID.	Precondition	Requirements
I06.FR.01	When the CSMS receives a TransactionEventRequest (<code>eventType = Updated</code>) from the Charging Station.	The CSMS SHALL check if there is updated tariff information available.
I06.FR.02	I06.FR.01 AND When there is updated tariff information available.	The CSMS SHALL respond with a TransactionEventResponse message to the Charging Station, containing the updated tariff information in the PersonalMessage field.
I06.FR.03	I06.FR.02	The Charging Station SHALL display the updated tariff information to the EV Driver.

J. MeterValues

1. Introduction

This Functional Block describes the functionality that enables a Charging Station to send periodic, possibly clock-aligned MeterValues.

The transfer of the MeterValues from the Charging Station to the CSMS will be taken over by the new Device Management Monitoring feature, however this mechanism has not been proven in the field yet. So the old [MeterValuesRequest](#) message remains available for use for now.

Extensive metering data relating to transactions can be recorded and transmitted in different ways depending on its intended purpose. There are two obvious use cases (but the use of meter values is not limited to these two):

- [Transaction Meter Values](#)
- [Clock-Aligned Meter Values](#)

Both types of meter readings MAY be reported in the *meterValue* element of the [TransactionEventRequest](#) message. [Clock-Aligned Meter Values](#) MAY be reported in standalone [MeterValuesRequest](#) messages.

2. Configuration

This section is normative.

2.1. Transaction Meter Values

Frequent (e.g. 1-5 minute interval) meter readings taken and transmitted (usually in "real time") to the CSMS, to allow it to provide information updates to the EV user (who is usually not at the Charging Station), via web, app, SMS, etc., as to the progress of the transaction. In OCPP, this is called "sampled meter data", as the exact frequency and time of readings is not very significant, as long as it is "frequent enough". "Sampled meter data" can be configured with the following Configuration Variables:

- [SampledDataTxStartedMeasurands](#)
- [SampledDataTxUpdatedMeasurands](#)
- [SampledDataTxUpdatedInterval](#)
- [SampledDataTxEndedMeasurands](#)
- [SampledDataTxEndedInterval](#)

[SampledDataTxUpdatedInterval](#) is the time (in seconds) between sampling of metering (or other) data, intended to be transmitted by [TransactionEventRequest](#) ([eventType = Updated](#)) messages during a transaction. A value of "0" (numeric zero), by convention, is to be interpreted to mean that no sampled data should be transmitted.

[SampledDataTxEndedInterval](#) is the time (in seconds) between sampling of metering (or other) data, intended to be transmitted in the [TransactionEventRequest](#) ([eventType = Ended](#)) message.

[SampledDataTxStartedMeasurands](#) is a comma separated list that prescribes the set of measurands to be included in the [meterValues](#) field of a [TransactionEventRequest](#) ([eventType = Started](#)).

[SampledDataTxUpdatedMeasurands](#) is a comma separated list that prescribes the set of measurands to be included in the [meterValues](#) field of a [TransactionEventRequest](#) ([eventType = Updated](#)), every [SampledDataTxUpdatedInterval](#) seconds.

[SampledDataTxEndedMeasurands](#) is a comma separated list that prescribes the sampled measurands to be included in the [meterValues](#) field of a [TransactionEventRequest](#) ([eventType = Ended](#)), these measurands have to be taken every [SampledDataTxEndedInterval](#) seconds from the start of the transaction, and will only be sent in the [TransactionEventRequest](#) ([eventType = Ended](#)).

Care should be taken to ensure that the amount of measurands that is expected at the end of a transaction fits in one [TransactionEventRequest](#)([eventType=Ended](#)) message. Keep the number of measurands in [SampledDataTxEndedMeasurands](#) to a minimum and configure a large interval in [SampledDataTxEndedInterval](#) to keep the number of samples small.

NOTE Please note: *Transaction related* MeterValues are never transmitted in [MeterValuesRequest](#).

2.2. Clock-Aligned Meter Values

Grid Operator might require meter readings to be taken from fiscally certified energy meters, at specific Clock aligned times (usually every quarter hour, or half hour).

"Clock-Aligned Meter Values" can be configured with the following Configuration Variables:

- [AlignedDataMeasurands](#)
- [AlignedTimeInterval](#)
- [AlignedDataTxEndedMeasurands](#)
- [AlignedDataTxEndedInterval](#)
- [AlignedDataSendDuringIdle](#)

[AlignedTimeInterval](#) is the size of the clock-aligned data interval (in seconds). This defines the set of evenly spaced meter data aggregation intervals per day, starting at 00:00:00 (midnight), at which time the Charging Station should take measurements and send them to the CSMS in a [MeterValuesRequest](#) message. A value of "0" (numeric zero), by convention, is to be interpreted to mean that no clock-aligned data should be transmitted.

[AlignedDataTxEndedInterval](#) is the size of the clock-aligned data interval (in seconds). This defines the set of evenly spaced

meter data aggregation intervals per day, starting at 00:00:00 (midnight) intended to be transmitted in the [TransactionEventRequest](#) ([eventType = Ended](#)) message.

For example, a value of 900 (15 minutes) indicates that every day should be broken into 96 15-minute intervals, starting at 0:00 and then measured every 15 minutes: 0:15, 0:30, 0:45, 1:00, 1:15 etc.

[AlignedDataMeasurands](#) is a comma separated list that prescribes the set of measurands to be included in a [MeterValuesRequest](#) PDU, every [AlignedTimeInterval](#) seconds.

[AlignedDataTxEndedMeasurands](#) is a comma separated list that prescribes the set of clock-aligned periodic measurands to be included in the [meterValue](#) elements of [TransactionEventRequest](#) ([eventType = Ended](#)) PDU for every [AlignedDataTxEndedInterval](#) of the transaction.

[AlignedDataSendDuringIdle](#) can be used to only send clock aligned meter values when there are no ongoing transactions.

2.3. Multiple Locations/Phases

When a Charging Station can measure the same measurand on multiple locations or phases, all possible locations and/or phases SHALL be reported when configured in one of the relevant Configuration Variables.

For example: A Charging Station capable of measuring *Current.Import* on *Inlet* (all 3 phases) (grid connection) and *Outlet* (3 phases per EVSE on both its EVSEs). *Current.Import* is set in [AlignedDataMeasurands](#). [AlignedTimeInterval](#) is set to 900 (seconds). Then the Charging Station should send: (every 15 minutes)

- a [MeterValuesRequest](#) with: evseld = 0; with 3 SampledValue elements, one per phase with location = Inlet.
- a [MeterValuesRequest](#) with: evseld = 1; with 3 SampledValue elements, one per phase with location = Outlet.
- a [MeterValuesRequest](#) with: evseld = 2; with 3 SampledValue elements, one per phase with location = Outlet.

NOTE

When the configuration variable [SampledDataRegisterValuesWithoutPhases](#) has the value *true*, then meter values of measurand *Energy.Active.Import.Register* will only report the total energy over all phases without reporting the individual phase values.

2.4. Signed Meter Values

OCPP 2.0.1 supports signed meter values. When a Charging Station support signed meter values it can use the Configuration Variables [AlignedDataSignReadings](#) and [SampledDataSignReadings](#) to report this. The CSMS can then use this same variables to turn the use of signed meter values *on* or *off*.

When enabled the Charging Station shall put the signed meter value in the *SignedMeterValue* field of the [SampledValue](#).

3. Use cases & Requirements

3.1. MeterValues

J01 - Sending Meter Values not related to a transaction

Table 152. J01 - Sending Meter Values not related to a transaction

No.	Type	Description
1	Name	Sending Meter Values not related to a transaction
2	ID	J01
	Functional block	J. Meter Values
3	Objective(s)	To sample the electrical meter or other sensor/transducer hardware to provide information about the Charging Stations' Meter Values.
4	Description	The Charging Station samples the electrical meter or other sensor/transducer hardware to provide information about its Meter Values. Depending on configuration settings, the Charging Station will send Meter Values.
	Actors	Charging Station, CSMS
	Scenario description	<p>1. The Charging Station sends a MeterValuesRequest message, for offloading Meter Values to the CSMS.</p> <p>2. Upon receipt of a MeterValuesRequest message, the CSMS responds with a MeterValuesResponse message.</p>
5	Prerequisite(s)	The Charging Station is configured to send Meter values every XX seconds. No transaction is running.
6	Postcondition(s)	<p>Successful postcondition: n/a</p> <p>Failure postcondition: n/a</p>



Figure 92. Sequence Diagram: Sending Meter Values

7	Error handling	n/a
8	Remark(s)	<p>The phase field is not applicable to all Measurands.</p> <p>The phase rotation of a Connector relative to the grid connection can be derived by querying the PhaseRotation Configuration Variables of all components in the chain from grid connection up to Connector.</p> <p>The nature of each <code>sampledValue</code> is determined by the optional Measurand, context, location, unit and phase fields.</p> <p>The optional <code>SignedMeterValue</code> field can contain digitally signed binary meter value data.</p>

J01 - Sending Meter Values not related to a transaction - Requirements

Table 153. J01 - Requirements

ID	Precondition	Requirement definition	Note
J01.FR.01		The Charging Station MAY sample the energy meter (or other sensor/transducer hardware) to provide extra information about its Meter Values.	It is up to the Charging Station when it will send Meter Values. This can be configured using the SetVariablesRequest message to data acquisition intervals and specify data to be acquired & reported.
J01.FR.02		The MeterValuesRequest message SHALL contain the id of the EVSE from which samples were taken.	
J01.FR.03	J01.FR.02 AND The evselid is 0.	The MeterValuesRequest message SHALL be associated with the entire Charging Station.	
J01.FR.04	J01.FR.03 AND Measurand is energy related.	The sample SHALL be taken from the main energy meter.	
J01.FR.05	If all captured at the same point in time.	Each MeterValue element SHALL contain a timestamp.	
J01.FR.06	If all captured at the same point in time.	Each MeterValue(s) element SHALL contain a set of one or more individual SampledValue elements.	
J01.FR.07		The optional measurand field SHALL specify the type of value being measured/reported.	
J01.FR.08		The optional context field SHALL specify the reason/event triggering the reading.	
J01.FR.09		The optional location field SHALL specify where the measurement is taken.	(e.g. Inlet, Outlet).
J01.FR.10		The optional phase field SHALL specify to which phase or phases of the electric installation the value applies.	
J01.FR.11		The Charging Station SHALL report all phase number dependent values from the electrical meter (or grid connection when absent) point of view.	
J01.FR.13	When reporting phase rotation of a component	The Charging Station SHALL report the phase rotation relative to the grid connection	
J01.FR.14	When configured to send MeterValuesRequest , See: Meter Values - Configuration	The Charging Station SHALL send MeterValuesRequest messages to the CSMS as configured.	
J01.FR.15	J01.FR.14 AND Amount of measurands is too much for 1 MeterValuesRequest	The Charging Station MAY use multiple MeterValuesRequest messages to send all measurands.	
J01.FR.17		The timestamp of a MeterValue SHALL apply to all its SampledValues .	
J01.FR.18	When CSMS receives a MeterValuesRequest	CSMS SHALL respond with MeterValuesResponse .	Failing to respond with MeterValuesResponse might cause the Charging Station to try the same message again.
J01.FR.19	If AlignedDataSendDuringIdle is set to true for an EVSE AND the specified EVSE has an ongoing transaction.	The Charging Station SHALL stop sending the clock aligned meter values for this EVSE.	
J01.FR.20	If AlignedDataSendDuringIdle is set to true for a Charging Station AND the Charging Station has an ongoing transaction.	The Charging Station SHALL stop sending the clock aligned meter values for all EVSEs and the main power meter.	

ID	Precondition	Requirement definition	Note
J01.FR.21	<code>AlignedDataSignReadings</code> is <code>true</code>	The Charging Station SHALL retrieve signed meter values from components that support data signing and put them in the <code>signedMeterValue</code> field.	

J02 - Sending transaction related Meter Values

Table 154. J02 - Sending transaction related Meter Values

No.	Type	Description
1	Name	Sending transaction related Meter Values
2	ID	J02
	<i>Functional block</i>	J. Meter Values
3	Objective(s)	To sample the energy meter or other sensor/transducer hardware to provide information about the Charging Stations' transaction related Meter Values.
4	Description	The Charging Station samples the energy meter or other sensor/transducer hardware to provide information about its transaction related Meter Values. Depending on configuration settings, the Charging Station will send Meter Values during a transaction.
	Actors	Charging Station, CSMS
	Scenario description	<ol style="list-style-type: none"> 1. The Charging Station sends a TransactionEventRequest (<code>eventType = Updated</code>) message, for offloading Meter Values to the CSMS. 2. Upon receipt of a TransactionEventRequest message, the CSMS responds with a TransactionEventResponse message.
5	Prerequisite(s)	The Charging Station is configured to send Meter Values every XX seconds. A transaction is running.
6	Postcondition(s)	<p>Successful postcondition: n/a</p> <p>Failure postcondition: n/a</p>

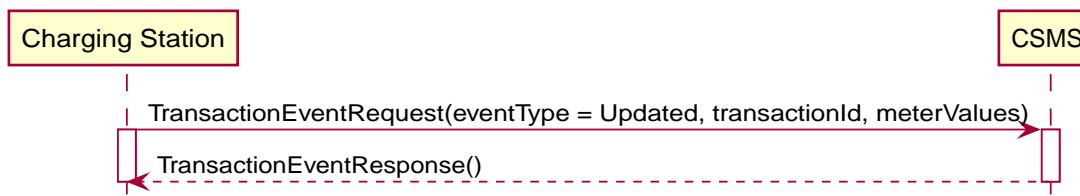


Figure 93. Sequence Diagram: Sending transaction related Meter Values

7	Error handling	When <i>Offline</i> , the Charging Station MUST queue any transaction-related messages (Meter Values belonging to a transaction) that it would have sent to the CSMS if the Charging Station had been online.
8	Remark(s)	<p>The phase field is not applicable to all Measurands.</p> <p>The phase rotation of a Connector relative to the grid connection can be derived by querying the PhaseRotation Configuration Variables of all components in the chain from grid connection up to Connector.</p> <p>The nature of each sampledValue is determined by the optional Measurand, context, location, unit and phase fields.</p> <p>The optional SignedMeterValue field can contain digitally signed binary meter value data.</p>

J02 - Sending transaction related Meter Values - Requirements

Table 155. J02 - Requirements

ID	Precondition	Requirement definition	Note
J02.FR.01		The Charging Station MAY sample the energy meter (or other sensor/transducer hardware) to provide extra information about its Meter Values.	It is up to the Charging Station when it will send Meter Values. This can be configured using the SetVariablesRequest message to data acquisition intervals and specify data to be acquired & reported.
J02.FR.02	If all captured at the same point in time.	Each MeterValue element SHALL contain a set of one or more individual SampledValue elements.	
J02.FR.03		The optional measurand field SHALL specify the type of value being measured/reported.	
J02.FR.04		The optional context field SHALL specify the reason/event triggering the reading.	
J02.FR.05		The optional location field SHALL specify where the measurement is taken (e.g. Inlet, Outlet).	
J02.FR.06		The optional phase field SHALL specify to which phase or phases of the electric installation the value applies.	
J02.FR.07		The Charging Station SHALL report all phase number dependent values from the power meter (or grid connection when absent) point of view.	
J02.FR.09	When reporting phase rotation of a component	The Charging Station SHALL report the phase rotation relative to the grid connection.	
J02.FR.10		The meterValue measurements in the same TransactionEventRequest message SHALL all belong to the timestamp in the message	meterValues for other timestamps should be sent in separate TransactionEventRequest messages.
J02.FR.11	When configured to send meter data in the TransactionEventRequest (<code>eventType = Updated</code>) AND When the interval in <code>SampledDataTxUpdatedInterval</code> has elapsed (See: Meter Values - Configuration)	The Charging Station SHALL send a TransactionEventRequest (<code>eventType = Updated</code>) with <code>triggerReason = MeterValuePeriodic</code> with the configured measurands in the <code>meterValue</code> field.	
J02.FR.12	J02.FR.11 AND <i>Offline</i> AND The Charging Station is running low on memory	The Charging Station MAY drop TransactionEventRequest (<code>eventType = Updated</code>) messages.	
J02.FR.13	J02.FR.12	When dropping TransactionEventRequest (<code>eventType = Updated</code>) messages, the Charging Station SHALL drop intermediate messages first (1st message, 3th message, 5th message etc.), not start dropping messages from the start or stop adding messages to the queue.	
J02.FR.14	J02.FR.11 AND Amount of meter data is too much for 1 TransactionEventRequest (<code>eventType = Updated</code>)	The Charging Station MAY use multiple TransactionEventRequest (<code>eventType = Updated</code>) messages with the same <code>timestamp</code> to send all measurands.	
J02.FR.16		All "Register" values relating to a single charging transaction, or a non-transactional consumer (e.g. Charging Station internal power supply, overall supply) MUST be monotonically increasing in time.	Except in the case of a meter replacement. See MeasurandEnumType .

ID	Precondition	Requirement definition	Note
J02.FR.17		For improved auditability, ".Register" values SHOULD be reported exactly as they are directly read from a non-volatile register in the electrical metering hardware, and SHOULD NOT be re-based to zero at the start of transactions	This allows any "missing energy" between sequential transactions, due to hardware fault, meter replacement, mis-wiring, fraud, etc. to be identified, by allowing the CSMS to confirm that the starting register value of any transaction is identical to the finishing register value of the preceding transaction on the same connector.
J02.FR.18		The timestamp of a MeterValue SHALL apply to all its SampledValues .	
J02.FR.19	When CSMS receives a TransactionEventRequest	CSMS SHALL respond with TransactionEventResponse .	Failing to respond with TransactionEventResponse might cause the Charging Station to try the same message again.
J02.FR.20	When configured to send meter data in the TransactionEventRequest (eventType = Ended) AND amount of meter data is too much for one TransactionEventRequest (eventType = Ended) message	Charging Station MAY remove samples until it fits in a message. When removing samples, the Charging Station SHOULD remove intermediate samples first (for example: 2nd sample, 4th sample, 6th sample etc.).	Samples should be removed in a way that it does not affect billing. See also E06.FR.12.
J02.FR.21	SampledDataSignReadings is true	The Charging Station SHALL retrieve signed meter values from components that support data signing and put them in the signedMeterValue field.	

3.2. ISO 15118 MeterValue signing

J03 - Charging Loop with metering information exchange

Table 156. J03 - Charging Loop with metering information exchange

No.	Type	Description
1	Name	Charging Loop with metering information exchange
2	ID	J03
	Functional block	J. Meter Values
	Reference	ISO15118-1 F1
3	Objectives	See ISO15118-1 , use case Objective F1, page 37.
4	Description	See ISO15118-1 , use case Description F1, page 37.
5	Prerequisites	<ul style="list-style-type: none"> - If authorization according use cases in Functional Block C is applied, it SHALL be finished successfully. See ISO15118-1 , use case Prerequisites F1, page 37.
6	Actors	EV, EVSE, Charging Station
7	Combined scenario description	15118 1a. The EV sends a ChargingStatusReq (in case of AC charging) message to the Charging Station, upon which EVSE returns a ChargingStatusRes containing the meter value from the fiscal meter. 1b. The EV sends a CurrentDemandReq (in case of DC charging) message to the Charging Station, upon which EVSE returns a CurrentDemandRes containing the meter value from the fiscal meter. 2. The EV sends a MeteringReceiptReq to the Charging Station to acknowledge receipt of the meter value.

No.	Type	Description
8	Postcondition(s)	See ISO15118-1, use case End conditions F1, page 37.

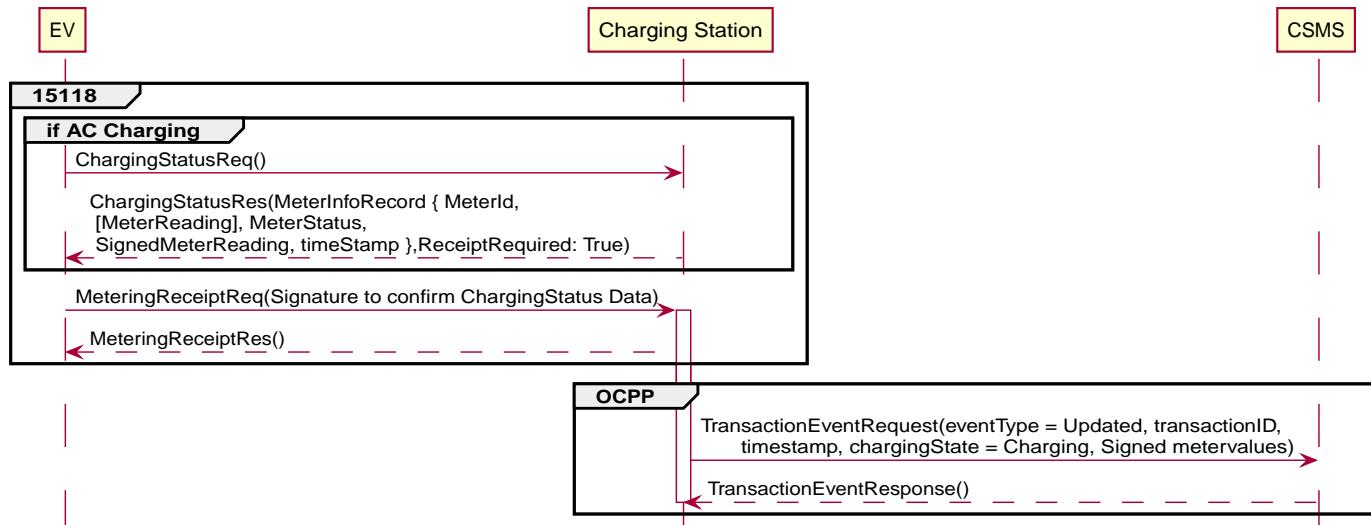


Figure 94. Charging Loop with metering information exchange

9	Error handling	n/a
10	Remark(s)	n/a

J03 - Charging Loop with metering information exchange - Requirements

Table 157. J03 - Requirements

ID	Precondition	Requirement definition	Note
J03.FR.04	When the Charging Station receives ISO 15118 signed MeteringReceiptReq message from EV	The Charging Station SHOULD NOT pass the meter value from the MeteringReceiptReq message to CSMS in a TransactionEventRequest (eventType = Updated) message. Instead, Charging Station sends transaction-related meter values as described in use case J02.	This does not imply that a Charging Station cannot require EV to send MeteringReceiptReq messages. An implementation at a Charging Station can be such, that every meter value from the fiscal meter that is send to CSMS (as per use case J02) must first have been acknowledged by a MeterReceiptReq from the EV.

K. SmartCharging

1. Introduction

This Functional Block describes all the functionalities that enable the CSO (or a third party) to influence the charging current/power transferred during a transaction, or set limits to the amount of current/power a Charging Station can draw from the grid.

Smart Charging in general has more than one definition. It can mean that the grid capacity is used in such a manner that consumers are able to charge their batteries fully at any time, even if large groups of consumers wish to 'fill up' simultaneously. Smart can also mean that energy prices can be taken into consideration when charging. Or again smart can be taken as using a local supply of sustainable energy from solar panels. And it is even 'smarter' when the Electric Vehicle (EV) driver wishes to be part of the solution. Within OCPP, Smart Charging means that a CSMS gains the ability to influence the (de-)charging power or current of a specific EV, or the total allowed energy consumption on an entire Charging Station / a group of Charging Stations. Different setups can be used. The following four typical kinds of smart charging will be used to illustrate the possible behavior of smart charging using OCPP:

- Internal Load Balancing
- Central Smart Charging
- Local Smart Charging
- External Smart Charging Control Signals

These types will be explained in [Types of Smart Charging](#). Of course, more complex use cases are possible in which two or more of the above use cases are combined into one more complex system.

NOTE

A mapping of the ISO 15118 and OCPP terminology is provided in [ISO 15118 and OCPP terminology mapping](#)

2. Types of Smart Charging

This section is informative.

2.1. Internal Load Balancing

The simplest form of smart charging is the Load Balancing use case. This concerns internal load balancing within the Charging Station, where the Charging Station controls current/power per EVSE. The Charging Station is configured with a fixed limit, e.g. the maximum current of the connection to the grid. The Charging Station in this case is responsible for optimizing charging for all its EVSEs. When a charging station is not directly connected to the grid, the energy system of a client will be responsible for the power supply.

This setup is typically used to set limits that are necessary due to known physical limits.

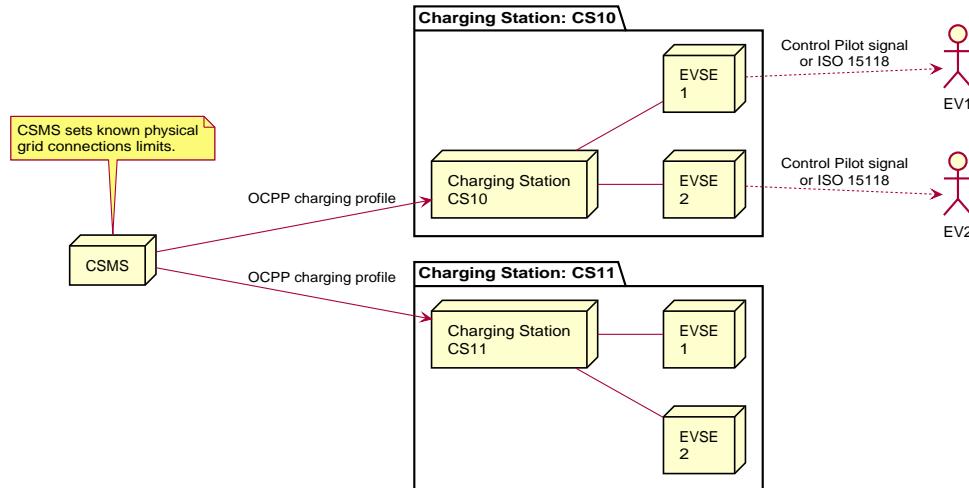


Figure 95. Internal Load Balancing Smart Charging Topology

2.2. Central Smart Charging

The next level in smart charging is when the CSMS has the ability to influence the charging power or current of a specific EV, the total allowed energy consumption on an entire Charging Station or a group of Charging Stations. Central Smart Charging assumes that charge limits are controlled by the CSMS. This could for example be based on a grid connection, energy availability on the grid (e.g. capacity forecast from the grid operator (DSO)) or the wiring of a building. In this setup, the CSMS can optimize charging not only on one Charging Station, but one level "up": it can optimize more than one Charging Station that share a connection and thus calculate a more efficient schedule for charging.

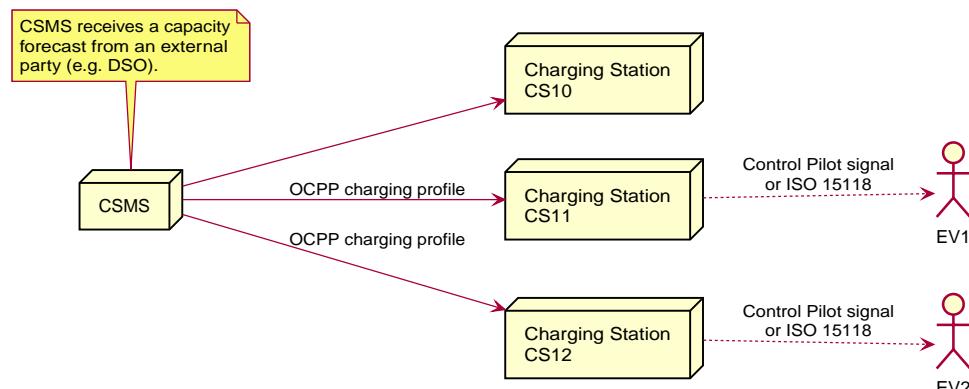


Figure 96. Central Smart Charging Topology

Central Smart Charging can be done with a Control Pilot signal, albeit with some limitations, because an EV cannot communicate its charging needs via the Control Pilot signal. In analogy to the [Local Smart Charging](#) use case, an EVSE can execute a charging schedule by the Control Pilot signal.

2.3. Local Smart Charging

Local Smart Charging describes a use case in which smart charging enabled Charging Stations have charging limits controlled locally by a Local Controller, not the CSMS. This type of smart charging assumes the existence of a Local Controller, which is a logical component that controls a group of Charging Stations. A typical use would be a number of Charging Stations in a parking garage where the rating of the connection to the grid is less than the sum of the ratings of the Charging Stations. Another application might be that the Local Controller receives information about the availability of power from a DSO or a local smart grid node.

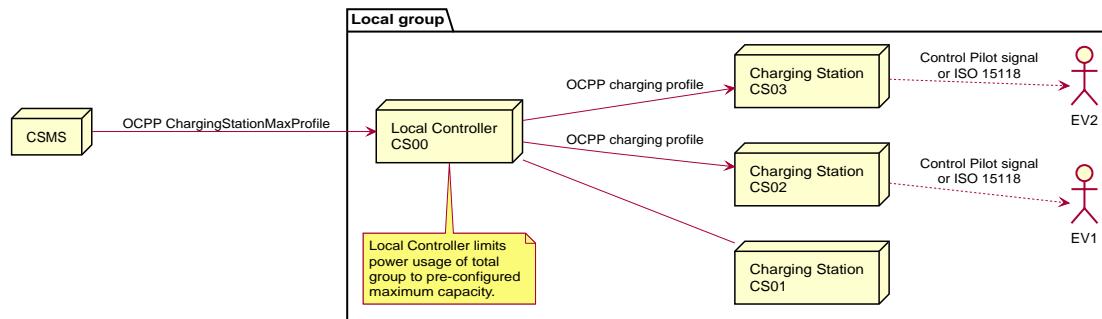


Figure 97. Local Smart Charging Topology

2.4. External Smart Charging Control Signals

The OCPP protocol is originally developed for communication between a CSMS and one or more Charging Stations. As described in the above, this means that a Charging Station Operator (CSO) CSMS controls a Charging Station and, based on the charging limits of both the EV and the Charging Station, the CSO determines how fast the EV is charged. However, in some situations / applications of OCPP enabled Charging Stations, these are not the only 2 factors that determine the charging speed. Other inputs that determine charging speed could be DSO signals (e.g. via IEC 61850 [[IEC61850-7-420](#)], IEC 60870 [[IEC60870-5-104](#)], DNP3 [[DNP3](#)] or OpenADR [[OPENADR](#)]) or signals from a Building / Home Energy Management System. Although these signals are out of scope for OCPP, it seems clear from an OCPP perspective that the CSMS is to be informed of changes in charging by external signals. However, this also leads to a number of questions, such as how to deal with conflicting signals. The figure below presents an example setup with an Energy Management System, where the external signals are visualized both in a setup with direct communication to the Charging Station as well as a multiple Charging Station setup using a Local Controller:

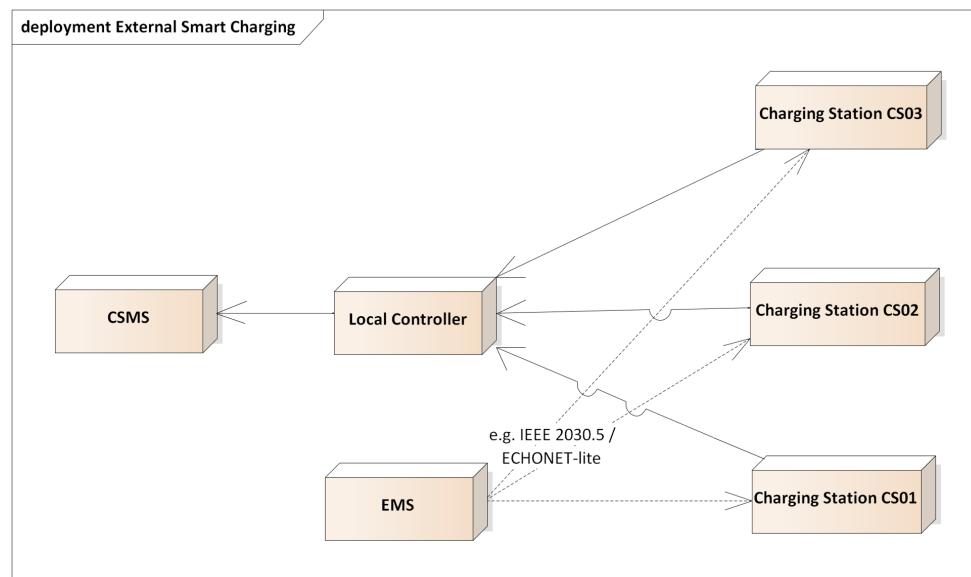


Figure 98. External Smart Charging

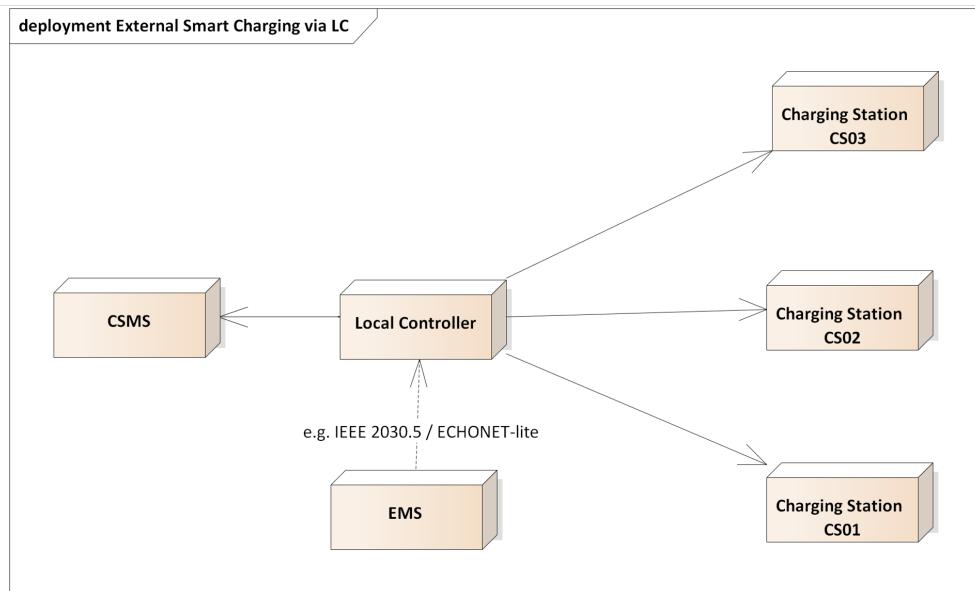


Figure 99. External Smart Charging via LC

If a Charging Station is connected both to the outside world as well as to an Energy Management System (EMS), this could result in a situation where the EMS, for whatever reason, decides that charging is not opportune, despite a charging schedule it might have received from the CSMS. This means that the Charging Station will not behave as expected by the CSMS. To prevent this, the Charging Station will have to be able to notify the CSMS that it has received a command from the EMS. An example reason could be an airconditioning system that is given preference / priority instead of charging an EV by a home user (in this case assuming that using the airconditioning and EV charging at the same time is not possible). This EMS might be in place to manage the maximum limit of a connection, but this can also be externally controlled.

3. Charging profiles

3.1. Introduction

Influencing the charge power or current is based on sending energy transfer limits at specific points in time to a Charging Station. Those limits are combined in a [ChargingProfile](#). A [ChargingProfile](#) holds the [ChargingSchedule](#) which defines a block of charging Power or Current limits and can contain a start time and duration. These can be applied to Charging Stations as well as to EVSEs of the Charging Stations. In [Example ChargingProfile](#) an example of a [ChargingProfile](#) is given to illustrate how these charging profiles can be used.

A CSMS can send a charging profile to a Charging Station using the message [SetChargingProfileRequest](#), in the following situations:

- At the start of a transaction to set the charging profile for the transaction
- In a RequestStartTransaction request sent to a Charging Station
- During a transaction to change the active profile for the transaction
- Outside the context of a transaction as a separate message to set a charging profile to a local controller, Charging Station, or a default charging profile to an EVSE.

3.2. Charging profile purposes

This section describes a number of types of charging profiles that are supported in OCPP. There are four different types of charging profiles, depending on their *purpose*:

ChargingProfile Purpose	Description
ChargingStationMaxProfile	In internal load balancing scenarios, the Charging Station has one or more local charging profiles that limit the power or current to be shared by all EVSEs of the Charging Station. The CSMS SHALL configure such a profile with ChargingProfilePurpose set to " <i>ChargingStationMaxProfile</i> ". <i>ChargingStationMaxProfile</i> can only be set at Charging Station evseld 0.
TxProfile	A transaction-specific profile with purpose <i>TxProfile</i> overrules the TxDefaultProfile for the duration of the current transaction only or until the TxProfile expires, whichever occurs earlier.
TxDefaultProfile	Default schedules for new transactions that MAY be used to impose charging policies. An example could be a policy that prevents charging during the day.
ChargingStationExternalConstraints	When an external system, not the CSMS, sets a charging limit or schedule, the Charging Station uses this purpose to report such a limit/schedule.

3.3. Charging profile recurrency

This section explains the different kinds of charging schedules that can be use in a charging profile, as defined by the value of the attribute *chargingProfileKind*:

ChargingProfile Kind	Description
Absolute	The charging schedule periods are relative to an absolute point in time defined in the schedule. This requires that <i>startSchedule</i> is set to a starting point in time. Use this, for example, to define a schedule that reduces charging between 17:00h and 21:00h, regardless of when charging session was started.
Recurring	The charging schedule restarts periodically at the first schedule period. To be most useful, this requires that <i>startSchedule</i> is set to a starting point in time. Use this in combination with <i>recurrencyKind = Daily</i> , for example, to define a schedule that reduces charging between 17:00h and 21:00h every day, regardless of when charging session was started.
Relative	Charging schedule periods start when ChargingProfile is activated. In most cases this will be at start of the power delivery. When a ChargingProfile is received for a transaction in progress, then it should activate immediately. No value for <i>startSchedule</i> should be supplied.

3.4. Stacking charging profiles

It is allowed to stack charging profiles of the same [ChargingProfile](#) purpose in order to describe complex calendars. For example, one can define [ChargingProfile](#) of purpose [TxDefaultProfile](#) with a duration and recurrence of one week that allows full power or current charging on weekdays from 23:00h to 06:00h and from 00:00h to 24:00h in weekends and reduced power or current charging at other times. On top of that, one can define other [TxDefaultProfiles](#) that define exceptions to this rule, for example for holidays.

A [ChargingProfile](#) holds a [ChargingSchedule](#) that defines limits for a certain time interval. Precedence of [ChargingSchedules](#) is determined by the [stackLevel](#) of their [ChargingProfile](#). When more than one [ChargingProfile](#) with the same [chargingProfilePurpose](#) is valid, then a [ChargingSchedule](#) of a [ChargingProfile](#) with a higher stack level overrules a [ChargingSchedule](#) from a [ChargingProfile](#) with a lower stack level.

To avoid conflicts, it is not allowed to have multiple charging profiles with the same [stackLevel](#) and same [chargingProfilePurpose](#) to be valid on the same EVSE at a given time. Note, that a charging profile for EVSE #0 is considered to be active on all EVSEs!

3.5. Combining Charging Profile Purposes

The Composite Schedule that will guide the charging level is a combination of the prevailing Charging Profiles of the different [chargingProfilePurposes](#) and stack levels.

As mentioned before, for each charging profile purpose, at any point in time, the leading charging schedule for that purpose is the charging schedule that has a schedule period defined for that time and that belongs to a charging profile with the highest stack level that is valid at that time, as determined by their [validFrom](#) and [validTo](#) parameters. The Composite Schedule is then calculated by taking the lowest charging limit (taking the different [chargingRateUnits](#) into account) among the leading profiles of the different purposes for each time interval.

The only exception is when both a [TxDefaultProfile](#) and a [TxProfile](#) are valid. In that case, the [TxProfile](#) will always overrule the [TxDefaultProfile](#), hence the Composite Schedule will not take the leading profile of purpose [TxDefaultProfile](#) into account in this specific situation. Note that time intervals do not have to be of fixed length, nor do they have to be the same for every [ChargingProfile](#) purpose. This means that a resulting Composite Schedule MAY contain intervals of different lengths.

In case the Charging Station is equipped with more than one EVSE, the limit value of [ChargingStationMaxProfile](#) is the limit for all EVSEs combined.

The two figures below will be used to give an example of combining multiple charging profiles with different stackLevels and Purposes.

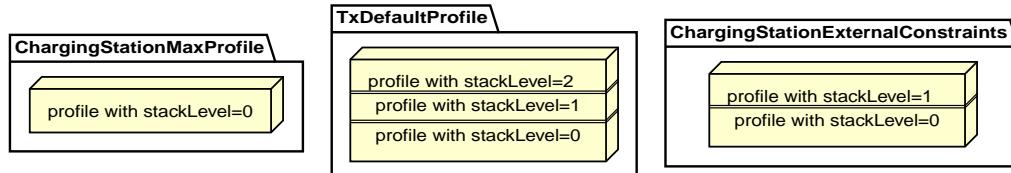


Figure 100. Multiple valid charging profiles - situation 1

Suppose that at a certain time interval the valid charging profiles are as in the above figure (situation 1). The composite schedule for this time interval will then be the lowest of the charging limits given in the [ChargingStationMaxProfile](#) with stackLevel 0, the [TxDefaultProfile](#) with stackLevel 2 and the [ChargingStationExternalConstraints](#) profile with stackLevel 1.

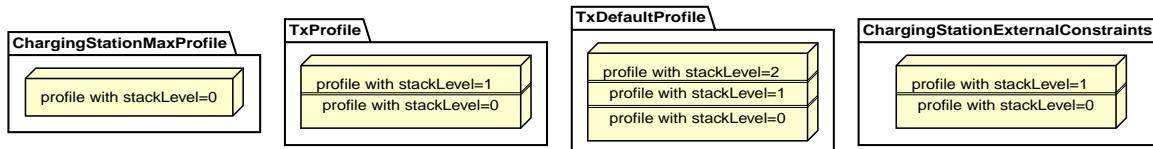


Figure 101. Multiple valid charging profiles - situation 2

On the other hand, consider the situation in which for a certain time interval the valid charging profiles are as in the above figure (situation 2). The composite schedule for this time interval will then be the lowest of the charging limits given in the [ChargingStationMaxProfile](#) with stackLevel 0, the [TxProfile](#) with stackLevel 1 and the [ChargingStationExternalConstraints](#) profile with stackLevel 1. Note that in this situation the [TxProfile](#) overrules the [TxDefaultProfile](#).

3.6. Example Charging Profile

This section is informative.

The following data structure describes a daily default profile that limits the power to 6 kW between 08:00h and 20:00h and to 11 kW between 00:00h and 08:00h and between 20:00h and 00:00h.

ChargingProfile			
chargingProfileId	100		
stackLevel	0		
chargingProfilePurpose	TxDefaultProfile		
chargingProfileKind	Recurring		
recurrencyKind	Daily		
chargingSchedule	(List of 1 ChargingSchedule elements)		
	ChargingSchedule		
	duration	86400 (= 24 hours)	
	startSchedule	2013-01-01T00:00Z	
	chargingRateUnit	W	
	chargingSchedulePeriod	(List of 3 ChargingSchedulePeriod elements)	
		ChargingSchedulePeriod	
		startPeriod	0 (=00:00)
		limit	11000
		numberPhases	3
		ChargingSchedulePeriod	
		startPeriod	28800 (=08:00)
		limit	6000
		numberPhases	3
		ChargingSchedulePeriod	
		startPeriod	72000 (=20:00)
		limit	11000
		numberPhases	3

IMPORTANT

The amount of phases used during charging is limited by the capabilities of: The Charging Station, EV and Cable between CS and EV. If any of these three is not capable of 3 phase charging, the EV will be charged using the number of phases that is supported by all three.

IMPORTANT

Switching the number of used phases during a schedule or transaction should be done with care. Some EVs MAY not support this and changing the amount of phases MAY result in physical damage. With the Configuration Variable: [Phases3to1](#) The Charging Station can tell if it supports switching the amount of phases during a transaction.

TIP

On days on which daylight saving goes into or out of effect, a special profile might be needed (e.g. for relative profiles).

3.6.1. Example Using Stacked Charging Profiles

A CSO wishes to limit charging to 2 kW during the peak hours of the day from 17:00h to 20:00h. This limit does not apply to Sundays and this limit does not apply to Christmas Day either.

If this applies to a large number of charging stations, then it is not practical to delete the charging profile every Sunday and then add it again on Monday. A possible solution is to add profiles with higher stack level for the exceptions to the base profile. See the following JSON examples where stack levels #2 and #3 are used to define exceptions for Sunday and Christmas.

(1) **TxDefaultProfile, stack #1:** time-of-day limitation to 2 kW, recurring every day from 17:00h to 20:00h.

```
"chargingProfile": {
  "id": 10, "stackLevel": 1, "chargingProfilePurpose": "TxDefaultProfile",
  "chargingProfileKind": "Recurring", "recurrencyKind": "Daily",
  "chargingSchedule": [ {
    "id": 1, "startSchedule": "2020-01-09T17:00:00", "duration": 1080,
    "chargingRateUnit": "W",
    "chargingSchedulePeriod": [ { "startPeriod": 0, "limit": 2000 } ]
  } ]
}
```

(2) **TxDefaultProfile, stack #2:** overruling Sundays to no limit, recurring every week starting 2020-01-05.

```
"chargingProfile": {
  "id": 11, "stackLevel": 2, "chargingProfilePurpose": "TxDefaultProfile",
  "chargingProfileKind": "Recurring", "recurrencyKind": "Weekly",
  "chargingSchedule": [ {
    "id": 1, "startSchedule": "2020-01-05T00:00:00", "duration": 86400,
    "chargingRateUnit": "W",
    "chargingSchedulePeriod": [ { "startPeriod": 0, "limit": 999999 } ]
  } ]
}
```

(3) **TxDefaultProfile, stack #3:** overruling Christmas Day 2020 to no limit, fixed date 2020-12-25.

Note, that this profile is only valid in the year 2020.

```
"chargingProfile": {
  "id": 12, "stackLevel": 3, "chargingProfilePurpose": "TxDefaultProfile",
  "chargingProfileKind": "Absolute",
  "validFrom": "2020-01-01T00:00:00", "validTo": "2021-01-01T00:00:00",
  "chargingSchedule": [ {
    "id": 1, "startSchedule": "2020-12-25T00:00:00", "duration": 86400,
    "chargingRateUnit": "W",
    "chargingSchedulePeriod": [ { "startPeriod": 0, "limit": 999999 } ]
  } ]
}
```

NOTE

Normally, when no limits are desired for charging, one will not define a charging schedule period for those hours (see stack level #1 for hours outside 17:00h - 20:00h). However, when overruling a charging schedule by one from a profile with a higher stack level, it is not possible to define a charging schedule period that has no limit. Therefore, the charging schedules for stack #2 and #3 in the above example use a (arbitrary) high value of 999999.

4. Smart Charging Signals to a Charging Station from Multiple Actors

This section is normative.

Within OCPP, multiple mechanism are supported for Smart Charging, i.e. multiple mechanisms are available that can add a limit when charging an EV:

1. The CSMS can influence charging by sending a SetChargingProfile message to the Charging Station. See [K01 - SetChargingProfile](#).
2. The EV can influence charging based on the PlugAndCharge functionality: the ISO 15118 enables EV initiated Charging Limits. See Section [5.3. ISO 15118 based Smart Charging](#).
3. Some local input, for example a Home Energy Management System (HEMS) or DSO, can influence the charging, for example via an External Smart Charging Control signal. See [K11 - Set / Update External Charging Limit](#).
4. A Charging Station can limit charging when it is load balancing when more than 1 EV is charging.

The assumption is that all parties that might be involved in setting limits for charging an EV will use one of the above mechanisms directly or indirectly.

To determine how a Charging Station should respond to simultaneous smart charging signals from multiple actors, the following rules should be followed:

Table 158. Smart Charging rules for multiple actor situation

ID	Precondition	Requirement definition	Note
SC.01		At any point in time, the charging limit, which is the result of merging the schedules from external sources and the OCPP charging profiles with the highest stackLevel from each of the purposes ChargingStationMaxProfile, ChargingStationExternalConstraints and TxDefaultProfile (or TxProfile), SHALL be less than or equal to the lowest value of available power or current in any of the merged schedules.	For safety purposes.
SC.02	When the ChargingProfile has changed	The Charging Station SHALL always inform the CSMS.	The message used for this varies depending on the which of the mechanisms mentioned at the start of this section is applicable: 1. n/a 2. NotifyEVChargingScheduleRequest 3. NotifyChargingLimitRequest 4. TransactionEventRequest
SC.03		Reporting to the CSMS concerning a changed limit in the ChargingProfile for mechanisms 3 and 4 as described in SC.02 MAY be skipped if the change in the limit is smaller than the percentage defined in the Configuration Variable: LimitChangeSignificance .	This is to prevent the Charging Station to send a lot of messages for small fluctuations (e.g. due to HEMS / smart meter input at the Charging Station)
SC.04		The GetCompositeScheduleResponse message SHALL always report the expected charging schedule, i.e. the lowest <i>limit</i> for charging. This means that when an EV has a charging limit X and indicates (e.g. using the ISO 15118 protocol) that it will use less energy than offered, amount Y, the Charging Station SHALL report limit Y.	

5. Use cases & Requirements

5.1. General Smart Charging

K01 - SetChargingProfile

Table 159. K01 - Central Smart Charging

No.	Type	Description
1	Name	SetChargingProfile
2	ID	K01
	Functional block	K. Smart Charging
3	Objective(s)	To enable the CSMS to influence the charging power or current drawn from a specific EVSE or the entire Charging Station over a period of time.
4	Description	The CSMS sends a SetChargingProfileRequest to the Charging Station to influence the power or current drawn by EVs. The CSMS calculates a ChargingSchedule to stay within certain limits, which MAY be imposed by any external system.
	Actors	Charging Station, CSMS, EV
	Scenario description	<ol style="list-style-type: none"> 1. The CSMS sets charging limits by sending SetChargingProfileRequest to the Charging Station. 2. The Charging Station responds with SetChargingProfileResponse.
5	Prerequisite(s)	n/a
6	Postcondition(s)	<p>Successful postcondition: The Charging Station <i>Successfully</i> influences the charging power or current of a specific EV, following the SetChargingProfileRequest sent by the CSMS.</p> <p>Failure postcondition: The Charging Station was <i>not able</i> to influence the charging power or current of a specific EV, following the SetChargingProfileRequest sent by the CSMS.</p>

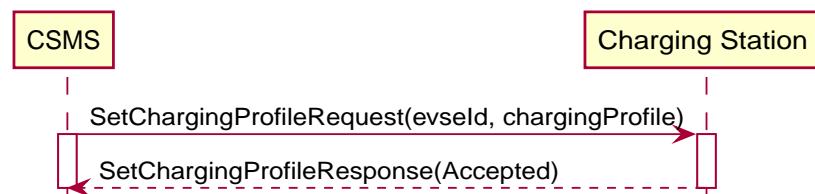


Figure 102. Sequence Diagram: SetChargingProfile

7	Error handling	n/a
8	Remark(s)	n/a

K01 - SetChargingProfile - Requirements

Table 160. K01 - Requirements

ID	Precondition	Requirement definition	Note
K01.FR.01		The CSMS MAY choose to set charging limits to a transaction using TxProfile .	
K01.FR.02		The CSMS MAY send a new charging profile for the EVSE that SHALL be used as a limit schedule for the EV.	
K01.FR.03		The CSMS SHALL include the <i>transactionId</i> in the SetChargingProfileRequest when setting a TxProfile .	The transactionId is used to match the profile to a specific transaction.
K01.FR.04	K01.FR.03 AND the given <i>transactionId</i> is known	The Charging Station SHALL apply the sent TxProfile to the transaction with the specified <i>transactionId</i> .	

ID	Precondition	Requirement definition	Note
K01.FR.05	When a SetChargingProfileRequest with an already known ChargingProfile.id is received AND the existing ChargingProfile does NOT have chargingProfilePurpose = ChargingStationExternalConstraints	The Charging Station SHALL replace the existing ChargingProfile with the one specified.	ChargingStationExternalConstraints profile cannot be replaced.
K01.FR.06	When chargingProfilePurpose is NOT TxProfile	The CSMS SHALL NOT send a ChargingProfile with a stackLevel - chargingProfilePurpose - evseld combination that already exists in another ChargingProfile (with different id) on the Charging Station and has an overlapping validity period.	This is to ensure that no two charging profiles with same stack level and purpose can be valid at the same time.
K01.FR.07	When the Charging Station accepts a SetChargingProfileRequest	The Charging Station SHALL re-evaluate its collection of charging profiles to determine which ChargingProfile will become active.	
K01.FR.08		The CSMS MAY send charging profiles to a Charging Station that are to be used as default charging profiles.	
K01.FR.09	When a SetChargingProfileRequest with a TxProfile is received AND there is no transaction active on the specified EVSE	The Charging Station SHALL send a SetChargingProfileResponse with status Rejected .	
K01.FR.10	When validFrom and validTo of a ChargingProfile are not set	The Charging Station SHALL consider the ChargingProfile to be valid indefinitely until it is explicitly replaced.	
K01.FR.11	If ChargingSchedule has a duration AND ChargingSchedulePeriod.startPeriod >= ChargingSchedule.duration	The Charging Station SHALL not execute the ChargingSchedulePeriod , because it is past the duration of the ChargingSchedule .	
K01.FR.12		A ChargingSchedulePeriod remains active until the next ChargingSchedulePeriod in the list starts or until ChargingSchedule.duration has elapsed.	
K01.FR.13	When recurrencyKind is used in combination with a ChargingSchedule duration shorter than recurrencyKind period.	The Charging Station SHALL fall back to default behavior after ChargingSchedule duration ends.	
K01.FR.14	When a SetChargingProfileRequest with a TxDefaultProfile and evseld = 0 is received AND No other TxDefaultProfile with the same stackLevel is installed on any specific EVSE.	The Charging Station SHALL apply, but not copy, this profile to all EVSEs.	A TxDefaultProfile charging profile on EVSE #0 is "owned by" EVSE #0, but has effect on all EVSEs.
K01.FR.15	When a SetChargingProfileRequest with a TxDefaultProfile and evseld > 0 is received AND No TxDefaultProfile with the same stackLevel is installed on EVSE #0.	The Charging Station SHALL only apply this profile to the specified EVSE.	
K01.FR.16		TxProfile SHALL only be used with evseld > 0 .	
K01.FR.17		When more than one ChargingProfile with the same chargingProfilePurpose is valid, as determined by their validFrom and validTo fields, then a ChargingSchedule from a ChargingProfile with a higher stackLevel overrules a ChargingSchedule from a ChargingProfile with a lower stackLevel .	

ID	Precondition	Requirement definition	Note
K01.FR.19		The CSMS SHALL NOT set phaseToUse in a SetChargingProfileRequest when numberPhases is other than 1.	
K01.FR.20		The CSMS SHALL NOT set phaseToUse in a SetChargingProfileRequest when the EVSE does not have ACPhaseSwitchingSupported defined and set to true.	
K01.FR.21		The optional ChargingSchedule field minChargingRate MAY be used by the Charging Station to optimize the power distribution between the EVSEs.	The parameter informs the Local Controller that charging below minChargingRate is inefficient, giving the possibility to select another balancing strategy.
K01.FR.22		The CSMS SHALL NOT set chargingProfilePurpose to ChargingStationExternalConstraints in a SetChargingProfileRequest .	This purpose is only used when an external system has set a charging limit/schedule.
K01.FR.26	When a SetChargingProfileRequest is received with a value for chargingRateUnit , that is not configured in the configuration variable ChargingScheduleChargingRateUnit .	Charging Station SHALL respond with SetChargingProfileResponse with status Rejected.	
K01.FR.27		ChargingProfiles set via SetChargingProfileRequest SHALL be persistent across reboots/power cycles.	
K01.FR.28	When a SetChargingProfileRequest is received for an evseld that does not exist.	Charging Station SHALL respond with SetChargingProfileResponse with status Rejected	
K01.FR.29	When Charging Station does not support smart charging.	Charging Station SHALL respond with RPC Framework CALLERROR: NotSupported.	
K01.FR.30	chargingProfile has a chargingSchedule with startSchedule set to a time in the future	The Charging Station SHALL only start imposing the limitation of this schedule as of point in time set by startSchedule	
K01.FR.31		The startPeriod of the first chargingSchedulePeriod in a chargingSchedule SHALL always be 0.	
K01.FR.32	(K01.FR.14 OR K01.FR.15) AND a transaction is active on the specified EVSE(s) (evseld = 0 refers to all EVSEs.)	The Charging Station SHALL continue the transaction on the specified EVSE(s), but switch to using the new/updated TxDefaultProfile .	
K01.FR.33	K01.FR.03 AND the given transactionId is not known	The Charging Station SHALL reject the SetChargingProfileRequest .	
K01.FR.34	The CSMS has not received a NotifyEVChargingNeedsRequest for the current transaction, i.e. charging session is not using ISO 15118	The ChargingProfile in the SetChargingProfileRequest SHALL contain only one ChargingScheduleType .	See use cases K15-K17 for ISO 15118 smart charging.
K01.FR.35		The list of ChargingSchedulePeriod elements in a chargingSchedule SHALL be ordered by increasing values of ChargingSchedulePeriod.startPeriod .	This means the list is in chronological order
K01.FR.36	When validFrom of a ChargingProfile is set	The Charging Station SHALL consider the ChargingProfile to be valid when current time \geq validFrom .	
K01.FR.37	When validTo of a ChargingProfile is set	The Charging Station SHALL consider the ChargingProfile to be valid when current time $<$ validTo .	

ID	Precondition	Requirement definition	Note
K01.FR.38	When <i>chargingProfilePurpose</i> = ChargingStationMaxProfile	<i>chargingProfileKind</i> SHALL NOT be Relative	
K01.FR.39	When <i>chargingProfilePurpose</i> is TxProfile	The CSMS SHALL NOT send a ChargingProfile with a <i>stackLevel - transactionId</i> combination that already exists in another ChargingProfile (with different <i>id</i>) with purpose TxProfile.	This is to ensure that no two charging profiles with same stack level and purpose can be valid at the same time.
K01.FR.40	When <i>chargingProfileKind</i> of a ChargingProfile is Absolute or Recurring	A value for <i>startSchedule</i> SHALL exist in the ChargingSchedule of the ChargingProfile .	This determines start date-time of the schedule and of the recurrence sequence.
K01.FR.41	When <i>chargingProfileKind</i> of a ChargingProfile is Relative	The field <i>startSchedule</i> SHALL be absent in the ChargingSchedule of the ChargingProfile .	A relative profile starts from when the profile is activated.
K01.FR.42	K01.FR.41	It is RECOMMENDED to make the ChargingSchedulePeriods relative to the moment the Charging Station is ready to deliver energy, i.e. when the EV driver is authorized and the EV is connected.	This is the point in a transaction where the charging station is ready to deliver energy. If PowerPathClosed is a TxStartPoint, then this will concur with the start of a transaction. In the next OCPP version, this will become a more strict requirement.
K01.FR.43	When a SetChargingProfileRequest with a value for <i>numberPhases</i> is received AND the EVSE is of type AC AND the Charging Station cannot ensure that no more than the received <i>numberPhases</i> will be used	The Charging Station SHALL respond with status = <i>Rejected</i>	Note that even when for example the ChargingProfile defines 3 phases and the Charging Station is able to charge with 3 phases, it is not guaranteed that the EV or cable are able to charge with 3 phases. Based on received MeterValues the CSMS can determine the used number of phases. Please refer to requirement K01.FR.50 and K01.FR.51, for correctly calculating the limits per phase.
K01.FR.44	When a SetChargingProfileRequest with a value for <i>numberPhases</i> or <i>phaseToUse</i> is received AND the EVSE is of type DC	The Charging Station MAY respond with status = <i>Accepted</i> , instead of <i>Rejected</i> and ignore the provided values for <i>numberPhases</i> and <i>phaseToUse</i> .	
K01.FR.45	When a SetChargingProfileRequest with a value for <i>numberPhases</i> is received AND the EVSE is of type AC AND the received <i>numberPhases</i> is NOT supported by the Charging Station and higher than the <i>numberPhases</i> that are supported by the Charging Station	The Charging Station MAY respond with status = <i>Accepted</i> , instead of <i>Rejected</i> and impose the limits to a lower <i>numberPhases</i>	Please refer to requirement K01.FR.50 and K01.FR.51, for correctly calculating the limits per phase.

ID	Precondition	Requirement definition	Note
K01.FR.46	When a SetChargingProfileRequest with <i>numberPhases</i> = 1 and a value for <i>phaseToUse</i> is received AND the EVSE is of type AC AND the EVSE is capable of switching the phase connected to the EV, which is indicated by ACPhaseSwitchingSupported defined as <i>true</i> OR the EVSE is already going to use the received <i>phaseToUse</i>	The Charging Station SHALL use the phase indicated by the received <i>phaseToUse</i> to connect to the EV.	
K01.FR.47	When a SetChargingProfileRequest with <i>numberPhases</i> = 1 and <i>phaseToUse</i> is omitted is received AND the EVSE is of type AC	The Charging Station SHALL select the phase on its own.	
K01.FR.48	When a SetChargingProfileRequest with a value for <i>phaseToUse</i> is received AND the EVSE is NOT capable of switching the phase connected to the EV, which is indicated by ACPhaseSwitchingSupported not being implemented or defined as <i>false</i> AND the EVSE is NOT going to use the received <i>phaseToUse</i>	The Charging Station SHALL respond with status = <i>Rejected</i> .	
K01.FR.49	When a SetChargingProfileRequest without a value for <i>numberPhases</i> is received AND the EVSE is of type AC	The Charging Station SHALL assume <i>numberPhases</i> = 3 as a default value.	
K01.FR.50	When a SetChargingProfileRequest with a chargingRateUnit = W is received AND The ChargingSchedule is used for AC charging	The Charging Station SHOULD calculate the phase current limit via: Current per phase = Power / (Line Voltage * Number of Phases).	The "Line Voltage" used in the calculation is not the measured voltage, but the set voltage for the area (for example, 230 or 110 V). The "Number of Phases" is the <i>numberPhases</i> from the ChargingSchedulePeriod. It is usually more convenient to use chargingRateUnit = A for AC charging.
K01.FR.51	When a SetChargingProfileRequest with a chargingRateUnit = A is received	The Charging Station SHALL use the provided limits, to limit the amount of Ampere per phase, not the sum of all phases.	

ID	Precondition	Requirement definition	Note
K01.FR.52	When a SetChargingProfileRequest with a TxDefaultProfile and evseld = 0 is received AND A TxDefaultProfile with the same stackLevel is installed on a specific EVSE and its chargingProfile.id does NOT equal the received chargingProfile.id	The Charging Station SHALL respond with a SetChargingProfileResponse with status Rejected and optionally with reasonCode = DuplicateProfile .	
K01.FR.53	When a SetChargingProfileRequest with a TxDefaultProfile and evseld > 0 is received AND A TxDefaultProfile with the same stackLevel is installed on EVSE #0 and its chargingProfile.id does NOT equal the received chargingProfile.id	The Charging Station SHALL respond with a SetChargingProfileResponse with status Rejected and optionally with reasonCode = DuplicateProfile .	

K02 - Central Smart Charging

Table 161. K02 - Central Smart Charging

No.	Type	Description
1	Name	Central Smart Charging
2	ID	K02
	<i>Functional block</i>	K. Smart Charging
3	Objective(s)	To enable the CSMS to influence the charging power or current drawn from a specific EVSE or the entire Charging Station over a period of time.
4	Description	<p>The CSMS sends a SetChargingProfileRequest to the Charging Station to influence the power or current drawn by the EV. The CSMS calculates a ChargingSchedule to stay within limits which MAY be imposed by any external system.</p> <p>See: Central Smart Charging</p>
	Actors	Charging Station, CSMS, EV, EV Driver
	<i>Scenario description</i>	<ol style="list-style-type: none"> After authorization the Charging Station will set a maximum current, that an EV might draw via the Control Pilot signal. This limit is based on (default) ChargingProfiles that the Charging Station previously received from the CSMS. The EV starts charging and a TransactionEventRequest is sent to the CSMS. The CSMS responds with a TransactionEventResponse. In response to a TransactionEventRequest the CSMS MAY choose to set charging limits to the transaction using a SetChargingProfileRequest. The Charging Station responds with a SetChargingProfileResponse. While charging is in progress the EVSE will continuously adapt the maximum current or power according to the installed ChargingProfiles.
	<i>Alternative scenario(s)</i>	K03 - Local Smart Charging K04 - Internal Load Balancing
5	Prerequisite(s)	The Functional Block <i>Smart Charging</i> is installed.
6	Postcondition(s)	<p>Successful postcondition: The Charging Station <i>Successfully</i> influences the charging power or current of a specific EV, following the SetChargingProfileRequest sent by the CSMS.</p> <p>Failure postcondition: The Charging Station was <i>not able</i> to influence the charging power or current of a specific EV, following the SetChargingProfileRequest sent by the CSMS.</p>

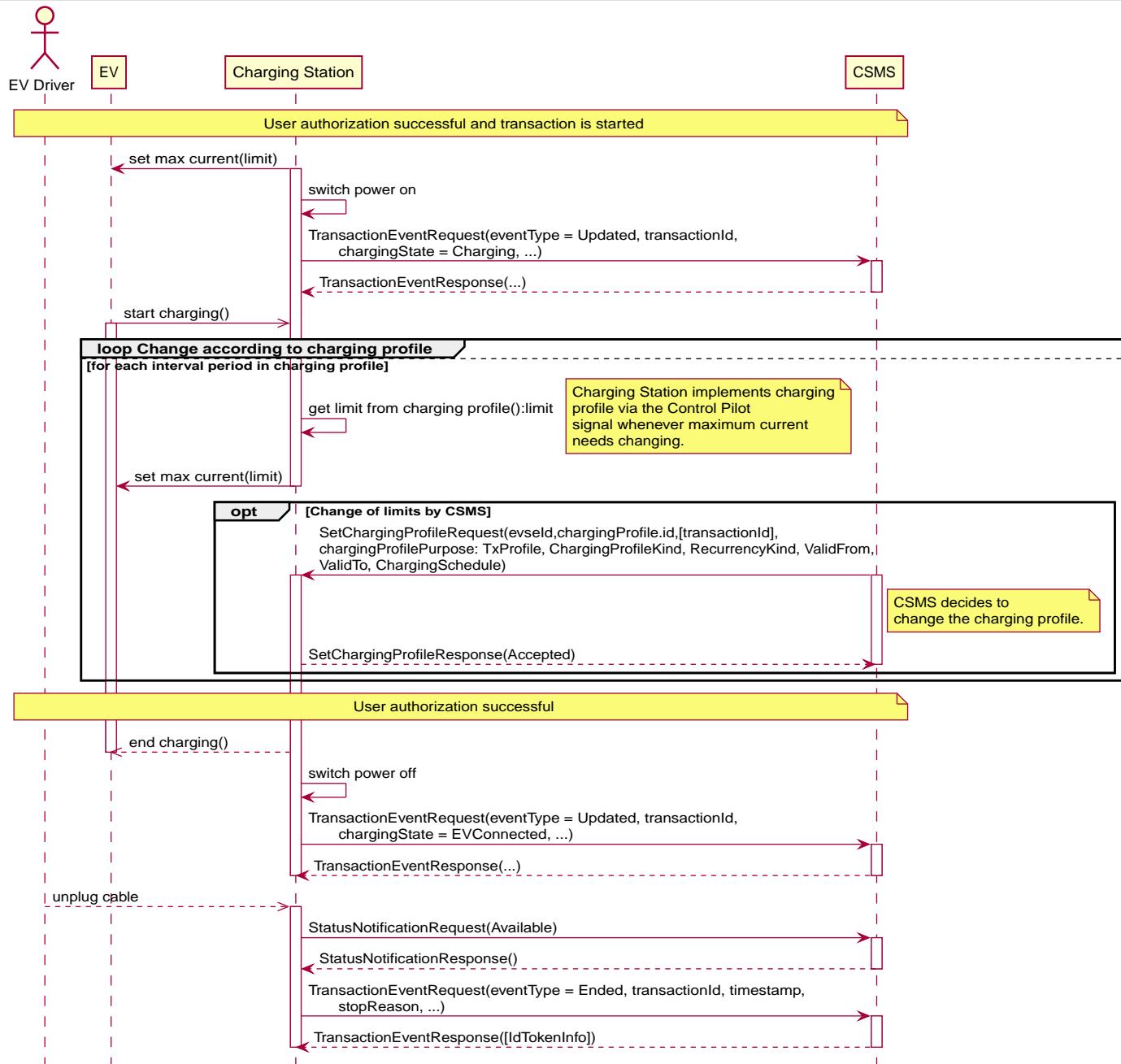


Figure 103. Sequence Diagram: Central Smart Charging

Explanation for the above figure:

- After authorization the EVSE will set a maximum current to use via the Control Pilot signal. This limit is based on a (default) charging profile that the EVSE had previously received from the CSMS. The EV starts charging and a [TransactionEventRequest](#) is sent to the CSMS.
- While charging is in progress the EVSE will continuously adapt the maximum current or power according to the charging profile. Optionally, at any point in time the CSMS may send a new charging profile for the EVSE. The Charging Station will then also take this new schedule into account when calculating a new composite schedule. This way the CSMS can influence the charging of an ongoing transaction.

7	Error handling	n/a
---	----------------	-----

8	Remark(s)	<p>The CSMS determines the constraints on ChargingSchedule per transaction.</p> <p>The CSMS imposes charging limits on EVSEs. In response to a TransactionEventRequest the CSMS may choose to set charging limits to the transaction using the TxProfile. It is RECOMMENDED to check the <i>offline</i> flag in TransactionEventRequest prior to sending a charging profile to check if the transaction is likely to be still ongoing, the TransactionEventRequest might have been cached during an <i>Offline</i> period.</p> <p>The final schedule constraints that apply to a transaction are determined by merging the profiles with purposes ChargingStationMaxProfile with the profile TxProfile or TxDefaultProfile in case no profile of purpose TxProfile is provided. Zero or more of the following ChargingProfile purposes MAY have been previously received from the CSMS: ChargingStationMaxProfile or TxDefaultProfile.</p> <p>It is recommended to omit the duration field of the ChargingSchedule from a TxProfile, so that it automatically lasts until the end of the transaction. If the TxProfile expires before the transaction ends, it falls back to the lowest limit of the active TxDefaultProfile and ChargingStationMaxProfile. If there are no other active profiles, it falls back to the local limit of the Charging Station.</p> <p>The scenario description and sequence diagram above are based on the Configuration Variable for start transaction being configured as follows:</p> <p>TxStartPoint: Authorized, DataSigned, PowerPathClosed, EnergyTransfer</p> <p>This use-case is also valid for other configurations, but then the transaction might start/stop at another moment, which might change the sequence in which message are send. For more details see the use case: E01 - Start Transaction options.</p>
---	------------------	--

K02 - Central Smart Charging - Requirements

Table 162. K02 - Requirements

ID	Precondition	Requirement definition	Note
K02.FR.01		The CSMS SHALL use charging profiles to stay within the limits imposed by any external system.	
K02.FR.02	After authorization.	The EVSE will set a maximum current to use via the Control Pilot signal.	This requirement only applies to AC chargers that use 61851. The limit may be based on a (default) charging profile that the EVSE previously received from the CSMS.
K02.FR.03		In order to ensure that an updated ChargingProfile applies only to the current transaction, the CSMS SHALL set the chargingProfilePurpose of the ChargingProfile to TxProfile .	An updated charging profile can be sent by the CSMS by sending a ChargingProfile with the same chargingProfileId .
K02.FR.04	If a transaction-specific profile with purpose TxProfile is present.	The TxProfile SHALL overrule the default charging profile with purpose TxDefaultProfile for the duration of the current transaction only.	
K02.FR.05	K02.FR.04 After the transaction is stopped	The TxProfile SHALL be deleted.	
K02.FR.06		The optional ChargingSchedule field minChargingRate MAY be used by the Charging Station to optimize the power distribution between the EVSEs.	The parameter informs the Local Controller that charging below minChargingRate is inefficient, giving the possibility to select another balancing strategy.
K02.FR.07		The CSMS SHALL NOT set chargingProfilePurpose to ChargingStationExternalConstraints in a SetChargingProfileRequest .	This purpose is only used when an external system has set a charging limit/schedule.

ID	Precondition	Requirement definition	Note
K02.FR.08	K02.FR.04 AND The charging schedule of TxProfile ends, before the transaction ends, because the set duration or validTo period expired	The Charging Station SHALL fall back to using the lowest limit of the active TxDefaultProfile and ChargingStationMaxProfile . If there are no other active profiles, it falls back to the local limit of the Charging Station	

K03 - Local Smart Charging

Table 163. K03 - Local Smart Charging

No.	Type	Description
1	Name	Local Smart Charging
2	ID	K03
	Functional block	K. Smart Charging
3	Objective(s)	To enable charging limits to be set at the Charging Station by a Local Controller.
4	Description	<p>Local Smart Charging describes a use case in which smart charging enabled Charging Stations have charging limits controlled locally by a Local Controller, not directly by the CSMS. The charging limits MAY either be pre-configured in the Local Controller in one way or another, or they can be set by the CSMS. The Local Controller SHALL contain the logic to distribute this capacity among the connected EVSEs by adjusting their limits as needed.</p> <p>This use case for Local Smart Charging is about limiting the amount of power that can be used by a group of Charging Stations, to a certain maximum.</p> <p>See Figure Local Smart Charging Topology</p>
	Actors	Charging Station, CSMS, EV, Local Controller, EV Driver
	Scenario description	<ol style="list-style-type: none"> 1. After authorization the Charging Station will set a maximum current, an EV might draw, via the Control Pilot signal. This limit is based on a TxDefaultProfile that the Charging Station previously received from the CSMS. 2. The EV starts charging, the Charging Station sends a TransactionEventRequest. 3. A TransactionEventRequest is sent to the CSMS via the Local Controller, so that the Local Controller knows a transaction has started. 4. During the transaction, the Local Controller sends a SetChargingProfileRequest to influence the charging current/power. 5. The Charging Station calculates the charging limits based on the installed ChargingProfiles. 6. The Local Controller just passes on the messages between Charging Station and CSMS, so that the CSMS can address all the Local Smart Charging group members individually. 7. While charging is in progress the EVSE will continuously adapt the maximum current according to the installed ChargingProfiles.
5	Prerequisite(s)	The Functional Block <i>Smart Charging</i> is installed.
6	Postcondition(s)	<p>Successful postcondition: The Local Controller <i>Successfully</i> controls maximum charging limits via the Control Pilot Signal.</p> <p>Failure postcondition: n/a</p>

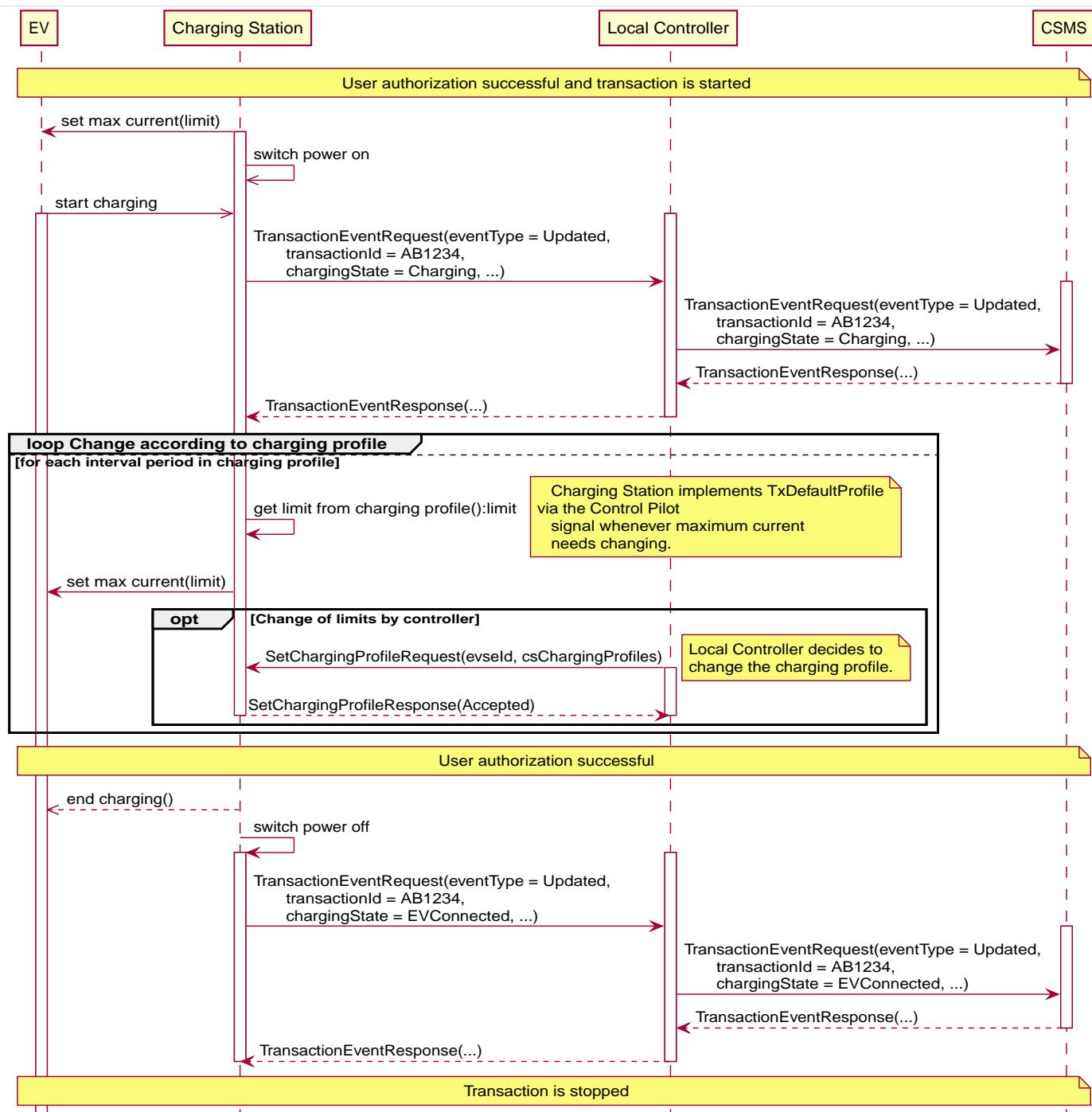


Figure 104. Sequence Diagram: Local Smart Charging

7	Error handling	n/a
8	Remark(s)	<p>The Local Controller for Local Smart Charging can be implemented in different ways, for example: as a separate physical component or as part of a 'master' Charging Station controlling a number of other Charging Stations.</p> <p>The Local Controller MAY or MAY NOT have any EVSEs of its own.</p> <p>The limits on Charging Stations in a Local Smart Charging group can either be pre-configured in the Local Controller in one way or another, or they can be set by the CSMS. The Local Controller contains the logic to distribute this capacity among the connected EVSEs by adjusting their limits as needed.</p>

K03 - Local Smart Charging - Requirements

Table 164. K03 - Requirements

ID	Precondition	Requirement definition	Note
K03.FR.01		The Local Controller MAY impose charging limits on a Charging Station.	
K03.FR.02	K03.FR.01	These limits MAY be changed dynamically during the charging process in order to keep the power consumption of the group of Charging Stations within the group limits.	
K03.FR.03	If at any point in time the Local Controller sends a new ChargingProfile to an EVSE	The Charging Station SHALL take this new ChargingProfile into account when calculating a new composite schedule that it will use to charge the EV.	
K03.FR.04		A Transaction with a chargingPriority that is higher than other transactions SHALL be fulfilled as long as possible, even if other transactions have to be suspended.	
K03.FR.05	If a chargingPriority is given in a TransactionEventResponse that is different from the chargingPriority in the IdTokenInfo .	The chargingPriority from the TransactionEventResponse SHALL be used for this transaction and for this transaction only.	It shall therefore not be stored e.g. in the Authorization Cache.
K03.FR.06	When no chargingPriority is known.	The Transaction or IdToken SHALL be assumed to have chargingPriority 0.	
K03.FR.07		The optional ChargingSchedule field minChargingRate MAY be used by the Charging Station to optimize the power distribution between the EVSEs.	The parameter informs the Local Controller that charging below minChargingRate is inefficient, giving the possibility to select another balancing strategy.
K03.FR.08		The Local Controller SHALL NOT set chargingProfilePurpose to ChargingStationExternalConstraints in a SetChargingProfileRequest .	This purpose is only used when an external system has set a charging limit/schedule.

K04 - Internal Load Balancing

Table 165. K04 - Internal Load Balancing

No.	Type	Description
1	Name	Internal Load Balancing
2	ID	K04
	Functional block	K. Smart Charging
3	Objective(s)	To enable internal load balancing within the Charging Station and between EVSEs.
4	Description	<p>The Load Balancing use case is about internal load balancing within the Charging Station, where the Charging Station controls current/power per EVSE.</p> <p>The Charging Station is configured with a fixed limit, e.g. the maximum current of the connection to the grid.</p> <p>See K01 - Set Charging Profile</p>
	Actors	Charging Station, CSMS, EVSE
	Scenario description	<ol style="list-style-type: none"> 1. The CSMS sets known physical grid connection limits by sending a ChargingProfile. 2. The Charging Station controls current/power per EVSE. 3. The EVSE sends a Control Pilot signal to the EV.
5	Prerequisite(s)	The Functional Block <i>Smart Charging</i> is installed.
6	Postcondition(s)	<p>Successful postcondition: The Charging Station <i>Successfully</i> balances the current/power between the different EVSEs, based on what the CSMS is sending.</p> <p>Failure postcondition: ChargingProfile is <i>not Accepted</i>. Charging is possible, although the Charging Station will <i>not</i> adhere to the ChargingProfile.</p>
7	Error handling	n/a
8	Remark(s)	n/a

K04 - Internal Load Balancing - Requirements

Table 166. K04 - Requirements

ID	Precondition	Requirement definition	Note
K04.FR.01		The Charging Station SHALL control the ChargingSchedule per EVSE.	
K04.FR.02		The Charging Station SHALL be configured with a fixed limit.	e.g. the maximum current of the connection to the grid.
K04.FR.03		A ChargingProfile with the purpose ChargingStationMaxProfile can only be set at Charging Station EVSE with Id 0.	
K04.FR.04		The optional ChargingSchedule field minChargingRate MAY be used by the Charging Station to optimize the power distribution between the EVSEs.	The parameter informs the Local Controller that charging below minChargingRate is inefficient, giving the possibility to select another balancing strategy.
K04.FR.05		The combined energy flow of all EVSEs (and the Charging Station hardware itself) SHALL NOT be greater than the limit set by ChargingStationMaxProfile .	

K05 - Remote Start Transaction with Charging Profile

Table 167. K05 - Remote Start Transaction with Charging Profile

No.	Type	Description
1	Name	Remote Start Transaction with Charging Profile
2	ID	K05
	Functional block	K. Smart Charging
3	Objective(s)	To enable the CSMS to remotely start a transaction by directly including a ChargingProfile , in order to assure that the transaction will use the right ChargingProfile .
4	Description	This use case covers how the CSMS can remotely start a transaction with purpose TxProfile . This assures that the right TxProfile is used. Also, when the Charging Station goes <i>Offline</i> after receiving RequestStartTransactionRequest . This is also needed, as switching from three phase- to one phase charging is not always possible and the transaction needs to start at the right phase.
	Actors	Charging Station, CSMS, External Trigger
	Scenario description	<ol style="list-style-type: none"> 1. The CSMS requests a Charging Station to remotely start a transaction by sending a RequestStartTransactionRequest with a ChargingProfile with purpose TxProfile. 2. The Charging Station responds with a RequestStartTransactionResponse indicating that it is able to start the transaction and will use the ChargingProfile. 3. The Charging Station informs the CSMS that a transaction has started by sending a TransactionEventRequest (eventType = Started) message. 4. The transaction is started in the same way as described in E. Transaction. 5. The Charging Station sends a TransactionEventRequest (eventType = Updated) to inform the CSMS that it is charging. 6. The Charging Station continues the regular smart charging session, following the set ChargingProfiles.
5	Prerequisite(s)	The Functional Block <i>Smart Charging</i> is installed.
6	Postcondition(s)	<p>Successful postcondition: The Charging Station <i>Successfully</i> charges taking into account the provided ChargingProfile.</p> <p>Failure postcondition: The transaction is <i>not</i> started. The Charging Station <i>Unsuccessfully</i> charges taking into account the provided ChargingProfile.</p>

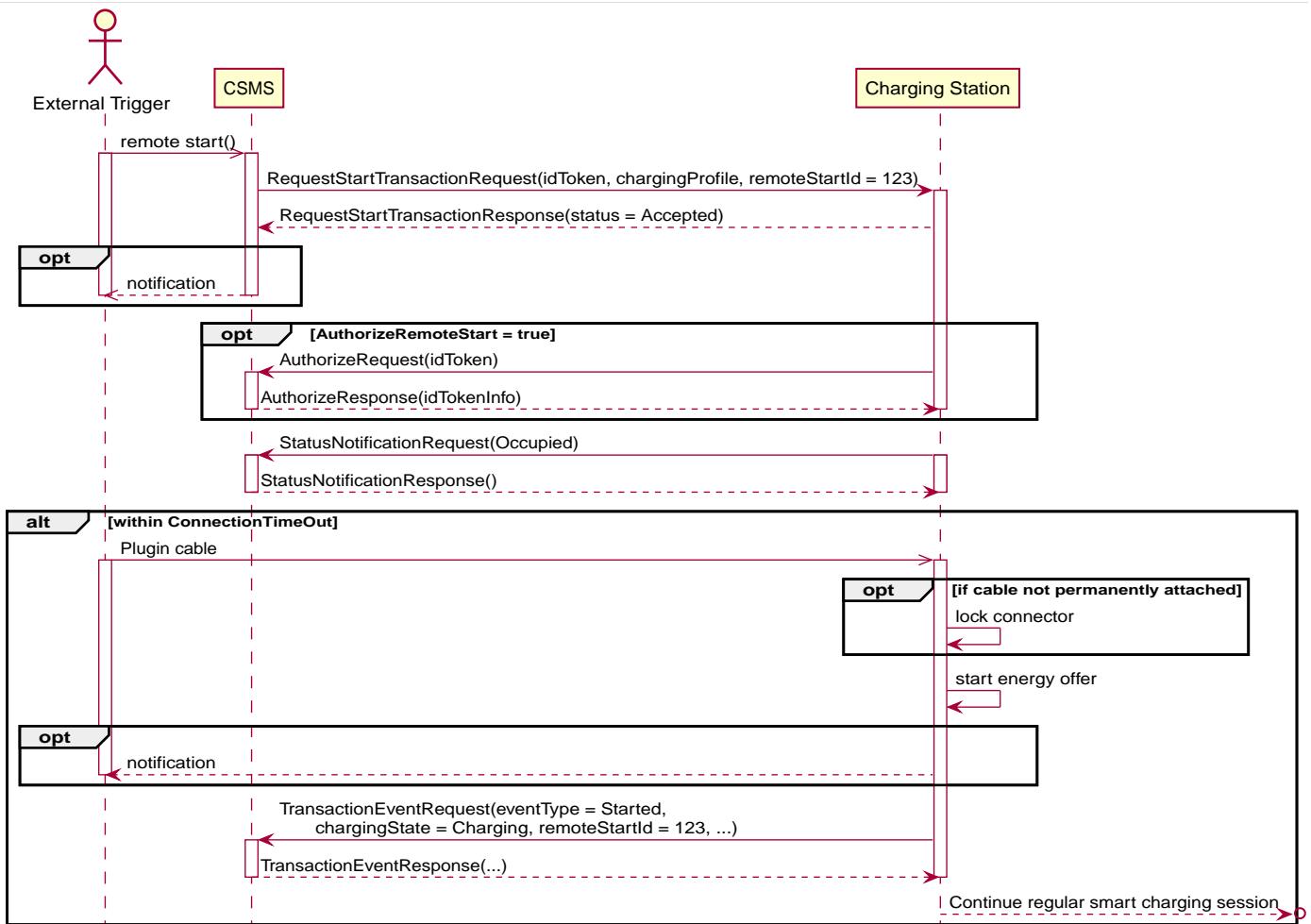


Figure 105. Sequence Diagram: Remote Start Transaction with Charging Profile

7	Error handling	n/a
8	Remark(s)	<p>The scenario description and sequence diagram above are based on the Configuration Variable for start transaction being configured as follows:</p> <p>TxStartPoint: EVConnected, Authorized, DataSigned, PowerPathClosed, EnergyTransfer</p> <p>This use-case is also valid for other configurations, but then the transaction might start/stop at another moment, which might change the sequence in which message are send. For more details see the use case: E01 - Start Transaction options.</p> <p>When a ChargingProfile with purpose TxProfile is provided as part of a RequestStartTransactionRequest, then a <i>transactionId</i> cannot be provided in the ChargingProfile, because it is not known at the time.</p>

K05 - Remote Start Transaction with Charging Profile - Requirements

Table 168. K05 - Requirements

ID	Precondition	Requirement definition	Note
K05.FR.01		The CSMS MAY include a ChargingProfile in a RequestStartTransactionRequest .	
K05.FR.02	K05.FR.01	The Purpose of the ChargingProfile SHALL always be TxProfile .	
K05.FR.03	K05.FR.01 AND NOT K05.FR.04	The Charging Station SHALL use the given profile to calculate its composite schedule.	
K05.FR.04	If a Charging Station without support for Smart Charging receives a RequestStartTransactionRequest with a ChargingProfile .	The Charging Station SHALL ignore the specified ChargingProfile .	The device model variable SmartChargingCtrlr.Enabled tells CSMS whether smart charging is supported.
K05.FR.05	If a Charging Station with support for Smart Charging receives a RequestStartTransactionRequest with an invalid ChargingProfile .	The Charging Station SHALL respond with RequestStartTransactionResponse with status = Rejected and optionally with reasonCode = "InvalidProfile" or "InvalidSchedule".	The device model variable SmartChargingCtrlr.Enabled tells CSMS whether smart charging is supported.

K06 - Offline Behavior Smart Charging During Transaction

Table 169. K06 - Offline Behavior Smart Charging During Transaction

No.	Type	Description
1	Name	Offline Behavior Smart Charging During Transaction
2	ID	K06
	Functional block	K. Smart Charging
3	Objective(s)	To enable the Charging Station to continue to use the current ChargingProfile for the duration of the transaction while it is <i>Offline</i> .
4	Description	If a Charging Station goes <i>Offline</i> after having received a transaction-specific ChargingProfile with purpose TxProfile , then it continues to use this profile for the duration of the transaction.
	Actors	Charging Station, CSMS, EV
	Scenario description	<ol style="list-style-type: none"> 1. The CSMS sends a SetChargingProfileRequest to the Charging Station with a TxProfile. 2. The Charging Station responds with a SetChargingProfileResponse. 3. While charging is in progress the EVSE will continuously adapt the maximum current or power according to the installed ChargingProfiles. 4. The Charging Station is <i>Offline</i> and operates stand-alone. 5. While charging is in progress the EVSE will continuously adapt the maximum current or power according to the already installed ChargingProfiles.
5	Prerequisite(s)	A transaction is ongoing. The Functional Block <i>Smart Charging</i> is installed.

No.	Type	Description
6	Postcondition(s)	<p>Successful postcondition: The Charging Station continues to use the charging profiles which are available.</p> <p>Failure postcondition: n/a</p>

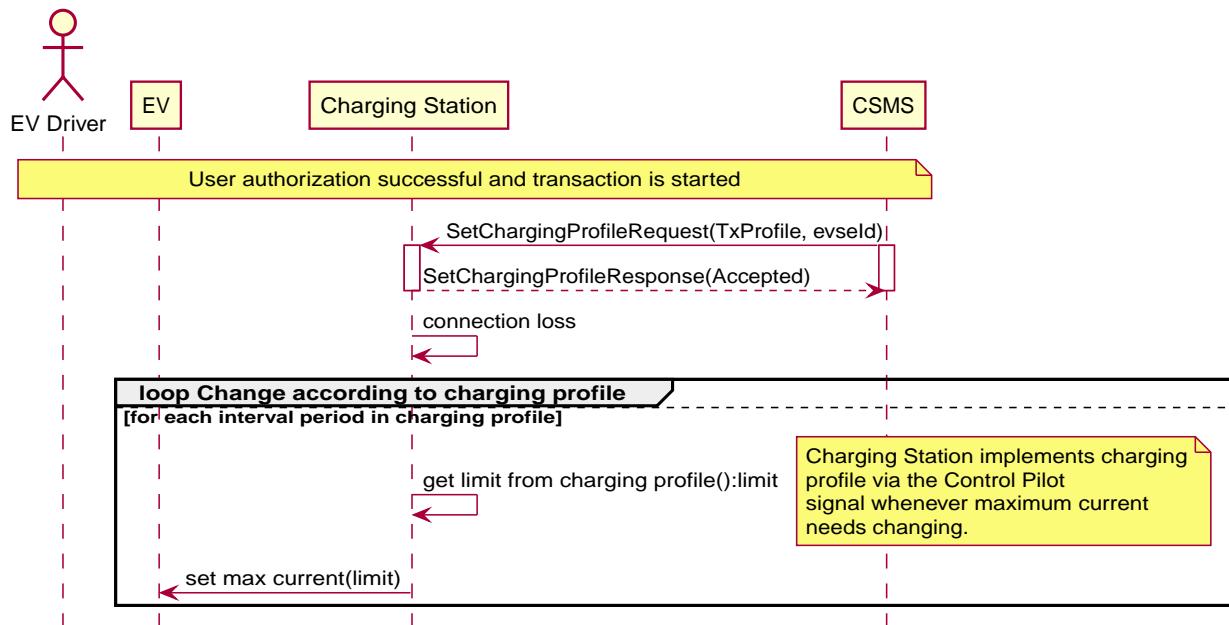


Figure 106. Sequence Diagram: Offline Behavior Smart Charging

7	Error handling	n/a
8	Remark(s)	n/a

K06 - Offline Behavior Smart Charging During Transaction - Requirements

Table 170. K06 - Requirements

ID	Precondition	Requirement definition
K06.FR.01	If the Charging Station goes <i>Offline</i> after having received a transaction-specific ChargingProfile with purpose TxProfile .	The Charging Station SHALL continue to use this profile for the duration of the transaction.
K06.FR.02	If the Charging Station goes <i>Offline</i> , without having any charging profiles.	The Charging Station SHALL execute the transaction as if no constraints apply.

K07 - Offline Behavior Smart Charging at Start of Transaction

Table 171. K07 - Offline Behavior Smart Charging at Start of Transaction

No.	Type	Description
1	Name	Offline Behavior Smart Charging at Start of Transaction
2	ID	K07
	Functional block	K. Smart Charging
3	Objective(s)	To enable the Charging Station to continue to use a ChargingProfile for a transaction which is started <i>Offline</i> .
4	Description	By setting a TxDefaultProfile on a Charging Station, the CSMS can assure that any transaction, which is started while the communication with the CSMS is <i>Offline</i> , uses this profile.
	Actors	Charging Station, CSMS, EV, EV Driver

No.	Type	Description
	Scenario description	<p>1. The CSMS sends a SetChargingProfileRequest to the Charging Station with a TxDefaultProfile.</p> <p>2. The Charging Station responds with a SetChargingProfileResponse.</p> <p>3. The Charging Station goes <i>Offline</i> and operates stand-alone.</p> <p>4. The Charging Station allows automatic authorization of any presented IdToken by either:</p> <ul style="list-style-type: none"> a. The Local Authorization List; a list of identifiers that can be synchronized with the CSMS. b. Authorization Cache entries; which autonomously maintains a record of previously presented identifiers that have been successfully authorized by the CSMS. (Successfully meaning: a response received on a message containing an IdToken). c. Configuration Variable: OfflineTxForUnknownIdEnabled = TRUE <p>5. The transaction is started in the same way as described in E. Transactions.</p> <p>6. While charging is in progress the EVSE will continuously adapt the maximum current or power according to the already installed ChargingProfiles.</p>
5	Prerequisite(s)	<p>The Charging Station is <i>Offline</i>.</p> <p>The Functional Block <i>Smart Charging</i> is installed.</p> <p>The IdToken is known in the Local Authorization List, the IdToken is known in the Authorization Cache, or unknown offline authorization is enabled.</p>
6	Postcondition(s)	<p>Successful postcondition: The Charging Station uses the installed TxDefaultProfile which are available for the <i>Offline</i> started transaction.</p> <p>Failure postcondition: n/a</p>

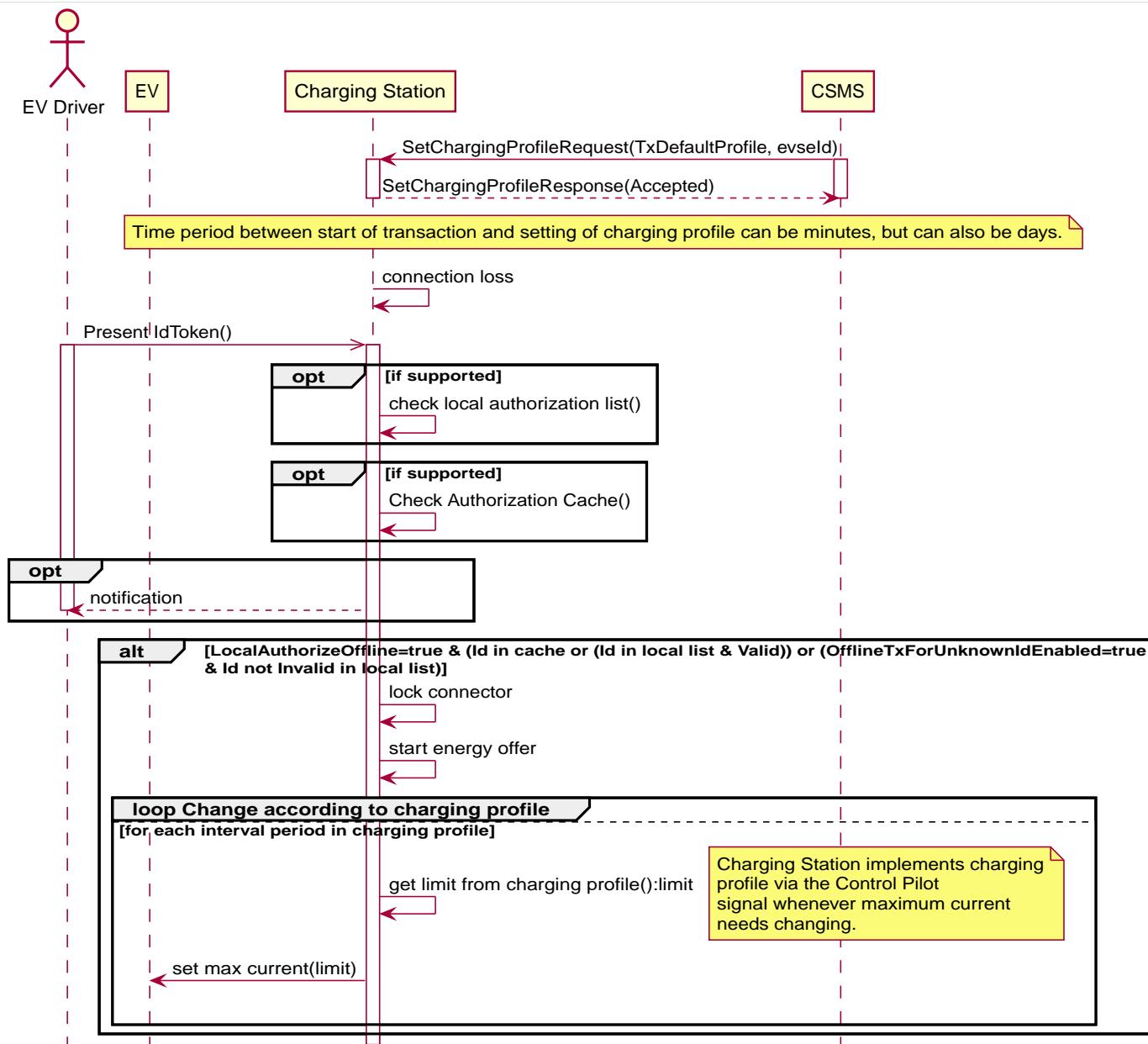


Figure 107. Sequence Diagram: Offline Behavior Smart Charging

7	Error handling	n/a
8	Remark(s)	See section Combining Charging Profile Purposes for a description on how to combine different charging profile purposes.

K07 - Offline Behavior Smart Charging at Start of Transaction - Requirements

Table 172. K07 - Requirements

ID	Precondition	Requirement definition	Note
K07.FR.01	If a Charging Station goes <i>Offline</i> before a transaction is started or before a transaction-specific ChargingProfile with purpose TxProfile was received.	The Charging Station SHALL use the charging profiles which are available.	With purpose TxDefaultProfile for the duration of the current transaction only.

K08 - Get Composite Schedule

Table 173. K08 - Get Composite Schedule

No.	Type	Description
1	Name	Get Composite Schedule
2	ID	K08
	Functional block	K. Smart Charging
3	Objective(s)	To request the Charging Station to report the composite charging schedule.
4	Description	<p>This use cases describes how the CSMS requests the Charging Station to report the Composite Charging Schedule, as calculated by the Charging Station, by sending GetCompositeScheduleRequest.</p> <p>The CompositeSchedule is the result of the calculation of all active schedules and possible local limits present in the Charging Station.</p>
	Actors	Charging Station, CSMS
	Scenario description	<ol style="list-style-type: none"> 1. The CSMS requests the Charging Station to report the Composite Charging Schedule by sending a GetCompositeScheduleRequest. 2. The Charging Station calculates the schedule. 3. The Charging Station responds with a GetCompositeScheduleResponse with the status and ChargingSchedule.
5	Prerequisite(s)	The Functional Block <i>Smart Charging</i> is installed.
6	Postcondition(s)	<p>Successful postcondition: The CSMS Successfully received the composite schedule from the Charging Station.</p> <p>Failure postcondition: The CSMS did <i>not</i> receive the composite schedule from the Charging Station.</p>

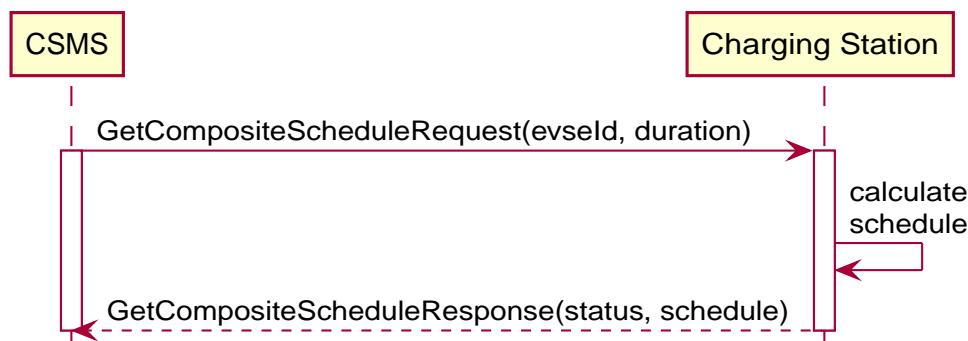


Figure 108. Sequence Diagram: Get Composite Schedule

7	Error handling	n/a
8	Remark(s)	<p>Please note that the charging schedule sent by the Charging Station is only indicative for that point in time. This schedule might change over time due to external causes (e.g. local balancing based on grid connection capacity is active and one EVSE becomes available).</p> <p>The Composite Schedule that will guide the charging level is a combination of the prevailing Charging Profiles of the different chargingProfilePurposes.</p> <p>This Composite Schedule is calculated by taking the minimum value for each time interval (see: Smart Charging signals to a Charging Station from multiple actors). Time intervals do not have to be of fixed length, nor do they have to be the same for every charging profile purpose. This means that a resulting Composite Schedule MAY contain intervals of different lengths.</p> <p>The reported schedule, in GetCompositeScheduleResponse, is the result of the calculation of all active schedules and possible local limits present in the Charging Station.</p> <p>The composite schedule reports the expected power or current the Charging Station expects to consume from the grid, for the requested EVSE, during the requested time period.</p> <p>When requested for evseid=0, the Charging Station will calculate the total expected consumption for the grid connection.</p>

K08 - Get Composite Schedule - Requirements

Table 174. K08 - Requirements

ID	Precondition	Requirement definition
K08.FR.01		The CSMS MAY request the Charging Station to report the CompositeSchedule by sending GetCompositeScheduleRequest .
K08.FR.02	Upon receipt of GetCompositeScheduleRequest .	The Charging Station SHALL calculate the scheduled time intervals, from the moment of message receipt up to the Duration (in seconds) and send them to the CSMS.
K08.FR.03	If the evselid in the GetCompositeScheduleRequest is set to '0'	The Charging Station SHALL report the total expected power or current the Charging Station expects to consume from the grid during the requested time period.
K08.FR.04		At any point in time, the available power or current in the CompositeSchedule , which is the result of merging the schedules of charging profiles ChargingStationMaxProfile , ChargingStationExternalConstraints and TxDefaultProfile (or TxProfile), SHALL be less than or equal to lowest value of available power or current in any of the merged schedules.
K08.FR.05	If the Charging Station is not able to report the requested schedule, for instance if the evselid is unknown	The Charging Station SHALL respond with the status Rejected .
K08.FR.06	K08.FR.02 AND When there is no transaction active on an EVSE	The Charging Station SHALL calculate the CompositeSchedule as if there is a transaction ongoing on the EVSE that is using the TxDefaultProfile (if this profile purpose is set)
K08.FR.07	When receiving a GetCompositeScheduleRequest with a chargingRateUnit , which is not configured in the configuration variable ChargingScheduleChargingRateUnit	The Charging Station SHALL respond with GetCompositeScheduleResponse with status Rejected .

K09 - Get Charging Profiles

Table 175. K09 - Get Charging Profiles

No.	Type	Description
1	Name	Get Charging Profile
2	ID	K09
	<i>Functional block</i>	K. Smart Charging
3	Objectives	To enable the CSMS to view the Charging Schedules/limits installed in a Charging Station, these can be installed by the CSMS or some other source.
4	Description	With the GetChargingProfilesRequest message the CSMS can ask a Charging Station to report all, or a subset of all the installed Charging Profiles from the different possible sources. This can be used for some automatic smart charging control system, or for debug purposes by a CSO.
	Actors	Charging Station, CSMS
	<i>Scenario description</i>	<p>1 The CSMS asks the Charging Station for the installed charging profiles by sending a GetChargingProfilesRequest message.</p> <p>2 The Charging Station responds, indicating if it can report Charging Schedules by sending a GetChargingProfilesResponse message.</p> <p>3 Charging Station sends a number of ReportChargingProfilesRequest messages to CSMS.</p> <p>4 The CSMS acknowledges reception of the reports by sending a ReportChargingProfilesResponse to the Charging Station for every ReportChargingProfilesRequest.</p>
5	Prerequisites	n/a
6	Postcondition(s)	The CSMS knows which charging profiles have been installed in the Charging Station that match the requested parameters.

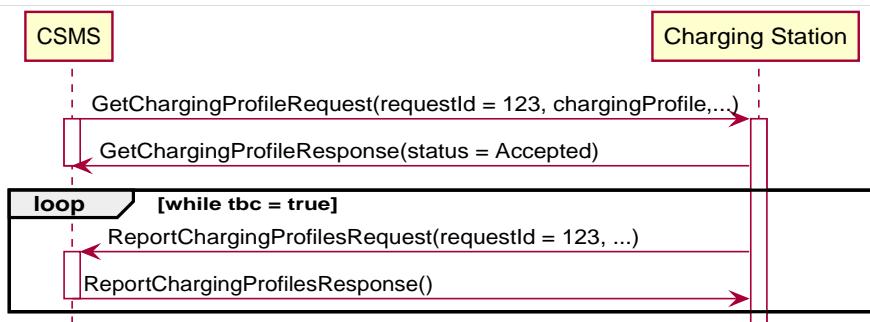


Figure 109. Sequence diagram of the use case "Get Charging Profiles"

7	Error Handling	When the Charging Station has no charging profiles that match the parameters in the GetChargingProfilesRequest the Charging Station SHALL respond with: NoProfiles .
8	Remarks	The charging profiles report can be split over multiple ReportChargingProfilesRequest messages, this can be because charging profiles for different charging sources need to be reported, or because there is just too much data for one message. To indicate that more reports will follow the flag tbc can be used.

K09 - Get Charging Profiles - Requirements

Table 176. K09 - Requirements

ID	Precondition	Requirements	Note
K09.FR.01	When <i>requestId</i> is set in the GetChargingProfilesRequest	The Charging Station SHALL set the <i>requestId</i> in every ReportChargingProfilesRequest that is sent as a result of this GetChargingProfilesRequest .	
K09.FR.02	When the charging profiles are reported in more than one ReportChargingProfilesRequest	The Charging Station SHALL set the <i>tbc</i> flag to true for all ReportChargingProfilesRequest messages except the last.	
K09.FR.03		The CSMS SHALL specify in <i>chargingProfile</i> criteria in GetChargingProfilesRequest either: - a (list of) <i>chargingProfileId</i> (s) OR - one or more of the fields <i>stackLevel</i> , <i>chargingLimitSource</i> , <i>chargingProfilePurpose</i> .	These fields are filter values of equal importance, but because a <i>chargingProfileId</i> uniquely identifies a charging profile, the other fields are not needed if <i>chargingProfileId</i> s are used.
K09.FR.04	If <i>evselId</i> is set to a value greater than 0 in the GetChargingProfilesRequest	The Charging Station SHALL report the installed charging profiles for the specified EVSE that match all fields in <i>chargingProfile</i> .	
K09.FR.05	If <i>evselId</i> is set to 0 in GetChargingProfilesRequest	The Charging Station SHALL only report charging profiles installed on the Charging Station itself (the grid connection) that match all fields in <i>chargingProfile</i> .	EVSE #0 can have a <i>ChargingStationMaxProfile</i> , <i>ChargingStationExternalConstraints</i> or a <i>TxDefaultProfile</i> . Note, that a <i>TxDefaultProfile</i> is not applied to EVSE #0 but to all individual EVSEs (see K01.FR.14).
K09.FR.06	If <i>evselId</i> is NOT set in the GetChargingProfilesRequest	The Charging Station SHALL report all installed charging profiles that match all fields in <i>chargingProfile</i> .	

K10 - Clear Charging Profile

Table 177. K10 - Clear Charging Profile

No.	Type	Description
1	Name	Clear Charging Profile

No.	Type	Description
2	ID	K10
	Functional block	K. Smart Charging
3	Objective(s)	To clear some or all of the charging profiles.
4	Description	If the CSMS wishes to clear some or all of the charging profiles that were previously sent to the Charging Station, then the CSMS sends a ClearChargingProfileRequest to the Charging Station.
	Actors	Charging Station, CSMS
	Scenario description	1. The CSMS sends a ClearChargingProfileRequest to the Charging Station. 2. The Charging Station responds with a ClearChargingProfileResponse specifying whether it was able to process the request in the status.
5	Prerequisite(s)	One or more ChargingProfiles are installed.
6	Postcondition(s)	Successful postcondition: The requested charging profiles are <i>Successfully</i> cleared. Failure postcondition: The requested charging profiles are <i>not</i> cleared, as no ChargingProfile is found.

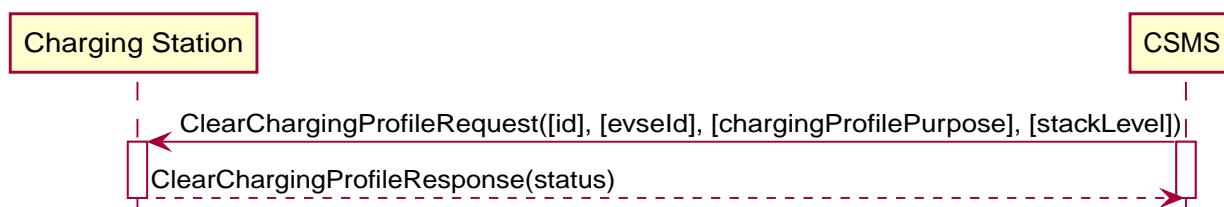


Figure 110. Sequence Diagram of the use case "Clear Charging Profile"

7	Error handling	n/a
8	Remark(s)	n/a

K10 - Clear Charging Profile - Requirements

Table 178. K10 - Requirements

ID	Precondition	Requirement definition	Note
K10.FR.01	If the Charging Station does not have any matching ChargingProfile .	Upon receipt of a ClearChargingProfileRequest , the Charging Station SHALL respond with the status <i>Unknown</i> .	
K10.FR.02		The CSMS SHALL either specify a <code>chargingProfile.id</code> OR include one or more of the fields <code>stackLevel</code> , <code>evsId</code> and <code>chargingProfilePurpose</code> in the ClearChargingProfileRequest to specify which Charging Profiles need to be cleared.	
K10.FR.03	Upon receipt of a ClearChargingProfileRequest with a specified <code>chargingProfileId</code> AND the <code>chargingProfilePurpose</code> of the referenced ChargingProfile is NOT <code>ChargingStationExternalConstraints</code>	The Charging Station SHALL clear the Charging Profile with the matching id and respond with a ClearChargingProfileResponse message with status = <i>Accepted</i> .	
K10.FR.04	NOT K10.FR.03 AND NOT K10.FR.08 AND Upon receipt of a ClearChargingProfileRequest , with optional values for <code>evsId</code> , <code>chargingProfilePurpose</code> , <code>stackLevel</code>	The Charging Station SHALL clear the ChargingProfile(s) that match (as logical AND) the values in the request, except those for that have <code>ChargingProfile = ChargingStationExternalConstraints</code> and respond with a ClearChargingProfileResponse message with status = <i>Accepted</i> .	

ID	Precondition	Requirement definition	Note
K10.FR.05	After clearing one or more Charging Profiles.	The Charging Station SHALL recalculate its composite schedule and set the resulting maximum power/current values to all ongoing transactions.	
K10.FR.06		The CSMS SHALL NOT set chargingProfilePurpose to ChargingStationExternalConstraints in a ClearChargingProfileRequest .	
K10.FR.07	K10.FR.05 AND the cleared profile has chargingProfilePurpose = TxDefaultProfile	The Charging Station SHALL continue any active transaction, that started with a TxDefaultProfile , as if it was started without a TxDefaultProfile .	
K10.FR.08	Upon receipt of a ClearChargingProfileRequest , with optional values for evseld , chargingProfilePurpose , stackLevel AND the matched ChargingProfile(s) all have ChargingProfile = ChargingStationExternalConstraints	The Charging Station SHALL respond with a ClearChargingProfileResponse message with status = Unknown.	Charging profiles for external constraints are disregarded by ClearChargingProfile message.
K10.FR.09	Upon receipt of a ClearChargingProfileRequest with a specified chargingProfileId AND the chargingProfilePurpose of the referenced ChargingProfile = ChargingStationExternalConstraints	The Charging Station SHALL respond with a ClearChargingProfileResponse message with status = Unknown.	Charging profiles for external constraints are disregarded by ClearChargingProfile message.

5.2. External Charging Limit based Smart Charging

K11 - Set / Update External Charging Limit With Ongoing Transaction

Table 179. K11 - Set / update external charging limit with ongoing transaction

No.	Type	Description
1	Name	Set / Update External Charging Limit With Ongoing Transaction
2	ID	K11
	Functional block	K. Smart Charging
3	Objectives	To inform the CSMS of a charging schedule or charging limit imposed by an External Control System on the Charging Station with ongoing transaction(s).
4	Description	An External Control System sends a charging limit/schedule to a Charging Station. This limit is sent to the CSMS.
	Actors	External Control System, Charging Station, CSMS
	Scenario description	<ol style="list-style-type: none"> 1. External control system sends charging limit/schedule to Charging Station. 2. Optional: Charging Station calculates new charging schedule. 3. Charging Station adjusts the charging speed of the ongoing transaction(s). 4. If the charging limit changed by more than: LimitChangeSignificance, the Charging Station sends a NotifyChargingLimitRequest message to CSMS with optionally the set charging limit/schedule. 5. The CSMS responds with NotifyChargingLimitResponse to the Charging Station. 6. If the charging rate changes by more than: LimitChangeSignificance, the Charging Station sends a TransactionEventRequest message to inform the CSMS. 7. The CSMS responds with TransactionEventResponse to the Charging Station.
5	Prerequisites	Charging Station is not in error state. An external system can set/clear a charging limit/schedule on the Charging Station via another connection than OCPP.
6	Postcondition(s)	The ongoing transaction will be limited by the received charging limit from the external system. The CSMS is informed of the new limit/schedule imposed by the external system.

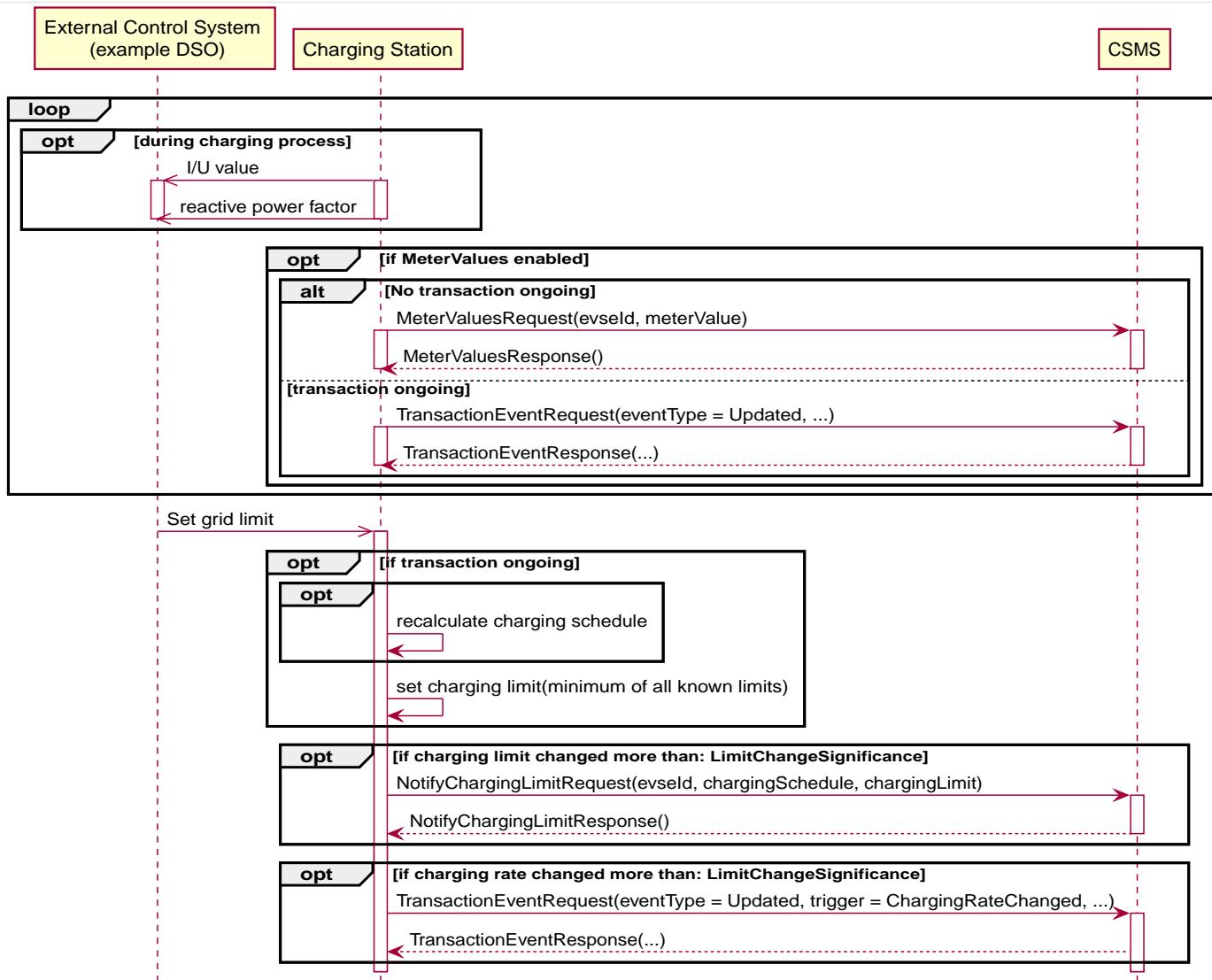


Figure 111. Sequence diagram of the use case "Setting / Updating External Charging Limit with Ongoing Transaction"

7	Error Handling	n/a
8	Remarks	The external system could, for example, use IEC 61850 [IEC61850-7-420] or OpenADR [OPENADR] to communicate the grid limit to the Charging Station, but this could be any protocol. Furthermore, an example of an external system is given, in this case a DSO that might set an external charging limit in case of grid problems, but this could be any other external system or reason to set a charging limit.

K11 - Set / Update External Charging Limit With Ongoing Transaction - Requirements

Table 180. K11 - Requirements

ID	Precondition	Requirements	Note
K11.FR.01	When an external charging limit/schedule is received during an ongoing transaction	The Charging Station SHALL NOT charge the ongoing transaction faster than this given limit/schedule.	
K11.FR.02	K11.FR.01 AND Charging limit changed by more than: LimitChangeSignificance	The Charging Station SHALL inform the CSMS of the new charging limit/schedule imposed by the external system by sending a NotifyChargingLimitRequest .	

ID	Precondition	Requirements	Note
K11.FR.03	K11.FR.02 AND EnableNotifyChargingLimitWithSchedules is true	The NotifyChargingLimitRequest SHALL contain the charging limits/schedules as set by the external system.	
K11.FR.04	K11.FR.01 AND Charging rate changed by more than: LimitChangeSignificance	The Charging Station SHALL send a TransactionEventRequest message to the CSMS with trigger = ChargingRateChanged	
K11.FR.05	K11.FR.02	The Charging Station SHALL NOT set the chargingLimitSource to CSO in the NotifyChargingLimitRequest .	
K11.FR.06	When an external charging limit/schedule is received	The Charging Station SHALL use purpose ChargingStationExternalConstraints when reporting about this limit (e.g. in a ReportChargingProfilesRequest).	It is RECOMMENDED to use negative values for the <i>id</i> of a ChargingStationExternalConstraints profile, to minimize the risk of a clash with an <i>id</i> that CSMS might use for a (future) charging profile.

K12 - Set / Update External Charging Limit Without Ongoing Transaction

Table 181. K12 - Set / update external charging limit without ongoing transaction

No.	Type	Description
1	Name	Set / Update External Charging Limit Without Ongoing Transaction
2	ID	K12
	<i>Functional block</i>	K. Smart Charging
3	Objectives	To inform the CSMS of a charging schedule or charging limit imposed by an external system on the Charging Station for new transactions or on the grid connection.
4	Description	An External Control System sends a charging limit to a Charging Station. This limit is sent to the CSMS.
	<i>Actors</i>	External Control System, Charging Station, CSMS
	<i>Scenario description</i>	<ol style="list-style-type: none"> 1. External Control System sends a charging limit to Charging Station (not during a transaction). 2. Optional: Charging Station calculates new charging schedule. 3. Charging Station adjusts the charging speed. 4. If the charging limit changed by more than: LimitChangeSignificance, the Charging Station sends a NotifyChargingLimitRequest message to CSMS with optionally the set charging limit/schedule. 5. The CSMS responds with a NotifyChargingLimitResponse to the Charging Station.
5	Prerequisites	Charging Station is not in error state. An external system that can set/clear a charging limit/schedule on the Charging Station via another connection than OCPP.
6	Postcondition(s)	New transactions will be limited by the received charging limit from the external system. The CSMS is informed of the new limit/schedule imposed by the external system.

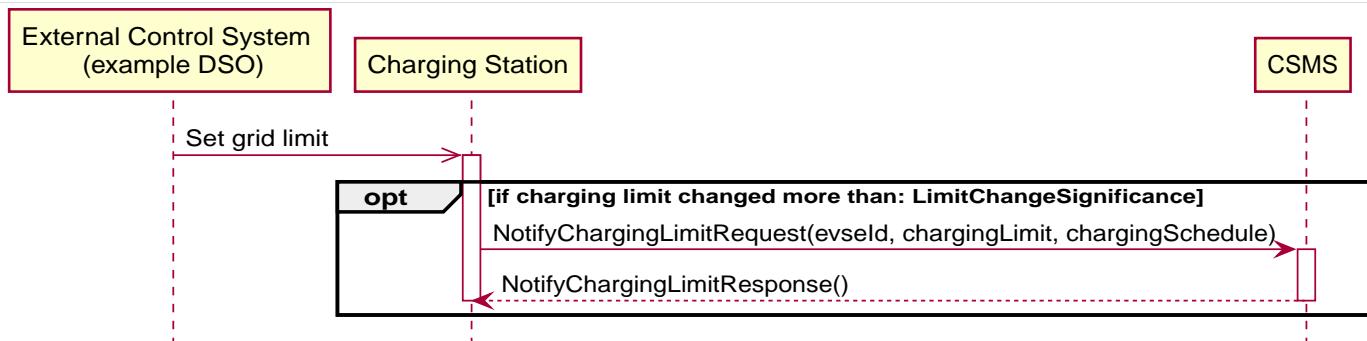


Figure 112. Sequence diagram of the use case "Set / Update External Charging Limit Without Ongoing Transaction"

7	Error Handling	n/a
8	Remarks	The external system could, for example, use IEC 61850 [IEC61850-7-420] or OpenADR [OPENADR] to communicate the grid limit to the Charging Station, but this could be any protocol. Furthermore, an example of an external system is given, in this case a DSO that might set an external charging limit in case of grid problems, but this could be any other external system or reason to set a charging limit.

K12 - Set / Update External Charging Limit Without Ongoing Transaction - Requirements

Table 182. K12 - Requirements

ID	Precondition	Requirements	Note
K12.FR.01	When an external charging limit/schedule is received while no transactions are ongoing	The total load of all EVSEs SHALL NOT exceed this given limit.	
K12.FR.02	K12.FR.01 AND Charging limit changed by more than: LimitChangeSignificance	The Charging Station SHALL inform the CSMS of the new charging limit/schedule imposed by the external system by sending a NotifyChargingLimitRequest .	
K12.FR.03	K12.FR.02 AND EnableNotifyChargingLimitWithSchedules is true	The NotifyChargingLimitRequest SHALL contain the charging limit/schedule as set by the external system.	
K12.FR.04	K12.FR.02	The Charging Station SHALL NOT set the chargingLimitSource to CSO in the NotifyChargingLimitRequest .	
K12.FR.05	When an external charging limit/schedule is received	The Charging Station SHALL use purpose ChargingStationExternalConstraints when reporting about this limit (e.g. in a ReportChargingProfilesRequest).	It is RECOMMENDED to use negative values for the id of a ChargingStationExternalConstraints profile, to minimize the risk of a clash with an id that CSMS might use for a (future) charging profile.

K13 - Reset / Release External Charging Limit

Table 183. K13 - Reset / Release External Charging Limit

No.	Type	Description
1	Name	Reset / Release External Charging Limit
2	ID	K13
	Functional block	K. Smart Charging

No.	Type	Description
3	Objectives	To release a charging limit that was previously imposed.
4	Description	An external control system sends a signal to release a previously imposed charging limit to a Charging Station. The Charging Station notifies the CSMS about this.
	Actors	External control system, Charging Station, CSMS
	Scenario description	<p>1. External control system release/removes a charging limit/schedule on the Charging Station</p> <p>2. When a transaction is ongoing, the Charging Station calculates the new Charging Schedule and adjusts charging speed.</p> <p>3. The Charging Station sends a ClearedChargingLimitRequest to notify the CSMS.</p> <p>4. The CSMS acknowledges with a ClearedChargingLimitResponse to the Charging Station.</p> <p>5. When the change has impact on an ongoing charging transaction and is more than: LimitChangeSignificance, the Charging Station sends a TransactionEventRequest to notify the CSMS.</p> <p>6. The CSMS acknowledges with a TransactionEventResponse to the Charging Station.</p>
5	Prerequisites	Previously, a charging limit was sent to the Charging Station under consideration. An external system that can set/clear a charging limit/schedule on the Charging Station via another connection than OCPP.
6	Postcondition(s)	The previously received charging limit is not limiting charging anymore.

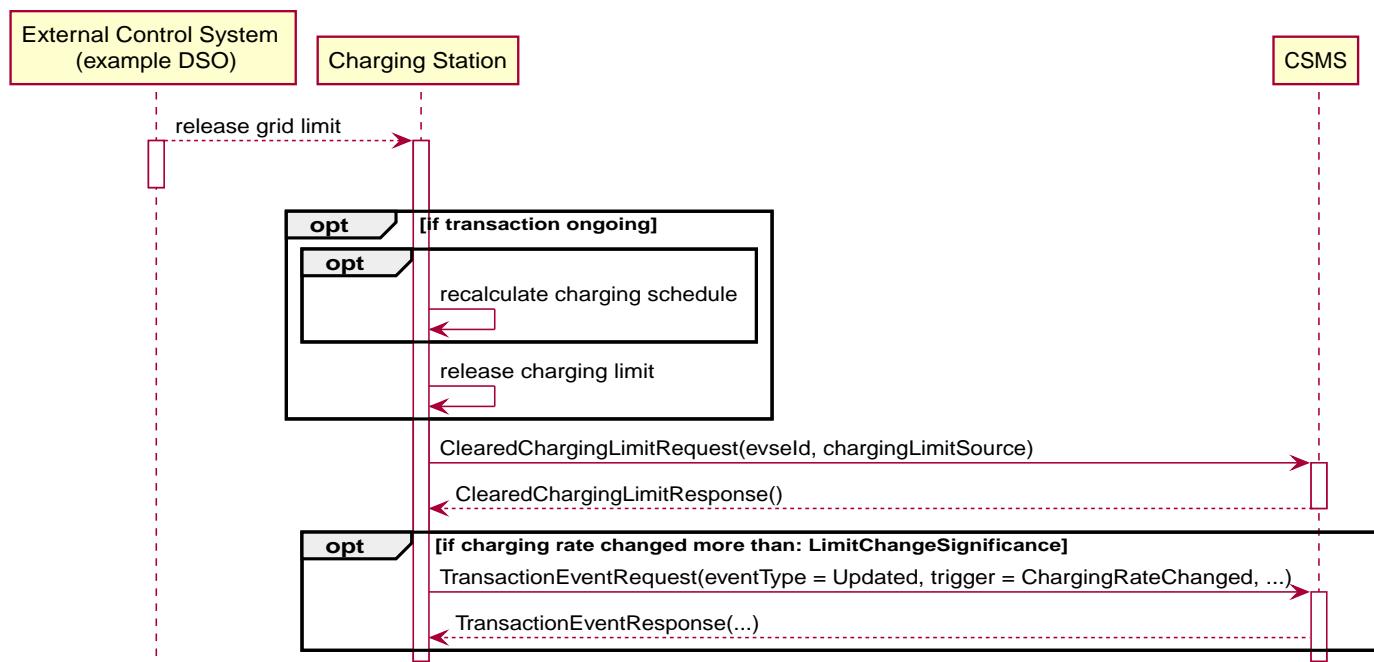


Figure 113. Sequence diagram of the use case "Release / Reset External Charging Limit"

7	Error Handling	n/a
8	Remarks	The external system could, for example, IEC 61850 [IEC61850-7-420] or OpenADR [OPENADR] to release the grid limit, but this could be any protocol. Furthermore, an example of an external system is given, in this case a DSO that might set an external charging limit in case of grid problems, but this could be any other external system or reason to set a charging limit.

K13 - Reset / Release External Charging Limit - Requirements

Table 184. K13 - Requirements

ID	Precondition	Requirements
K13.FR.01	A transaction is ongoing AND External charging limit is released/removed	The Charging Station SHALL NOT limit charging anymore based on the previously received limit.
K13.FR.02	K13.FR.01	The Charging Station SHALL notify the CSMS by sending a ClearedChargingLimitRequest message.

ID	Precondition	Requirements
K13.FR.03	K13.FR.01 AND Charging rate changed by more than: LimitChangeSignificance	The Charging Station SHALL send a TransactionEventRequest message to the CSMS with trigger = ChargingRateChanged .

K14 - External Charging Limit with Local Controller

Table 185. K14 - External Charging Limit with Local Controller

No.	Type	Description
1	Name	Handle external charging limit with a local controller
2	ID	K14
	Functional block	K. Smart Charging
3	Objective(s)	To adjust the charging limits according to the External Control System requirements.
4	Description	An external control system sends a charging limit to the Local Controller. The Local Controller notifies the CSMS, calculates the new charging schedules and sends a SetChargingProfileRequest messages to all Charging Stations for which the charging profile has changed.
	Actors	External control system, Local Controller, Charging Station, CSMS
	Scenario description	<ol style="list-style-type: none"> 1. External control system sends a charging limit/schedule to Local Controller. 2. Local Controller sends a NotifyChargingLimitRequest message to the CSMS. 3. Local Controller calculates new Charging Profiles for all connected Charging Stations. 4. Local Controller sends a SetChargingProfileRequest message to all Charging Stations for which the charging profile has changed. 5. External control system sends a charging limit/schedule to Local Controller. 6. Local Controller sends a ClearedChargingLimitRequest message to the CSMS. 7. Local Controller calculates new Charging Profiles for all connected Charging Stations. 8. Local Controller sends a ClearChargingProfileRequest messages to all affected Charging Stations.
5	Prerequisite(s)	Ongoing transaction(s). An external system that can set/clear a charging limit/schedule on Local Controller via another connection than OCPP.
6	Postcondition(s)	<p>Successful postcondition: The ongoing transactions will be limited by the received charging limit from the external system. The CSMS is informed of the new limit/schedule imposed by the external system.</p> <p>Failure postcondition: The CSMS is not informed about the changed charging limit. The External Control System is not able to change the charging limit.</p>

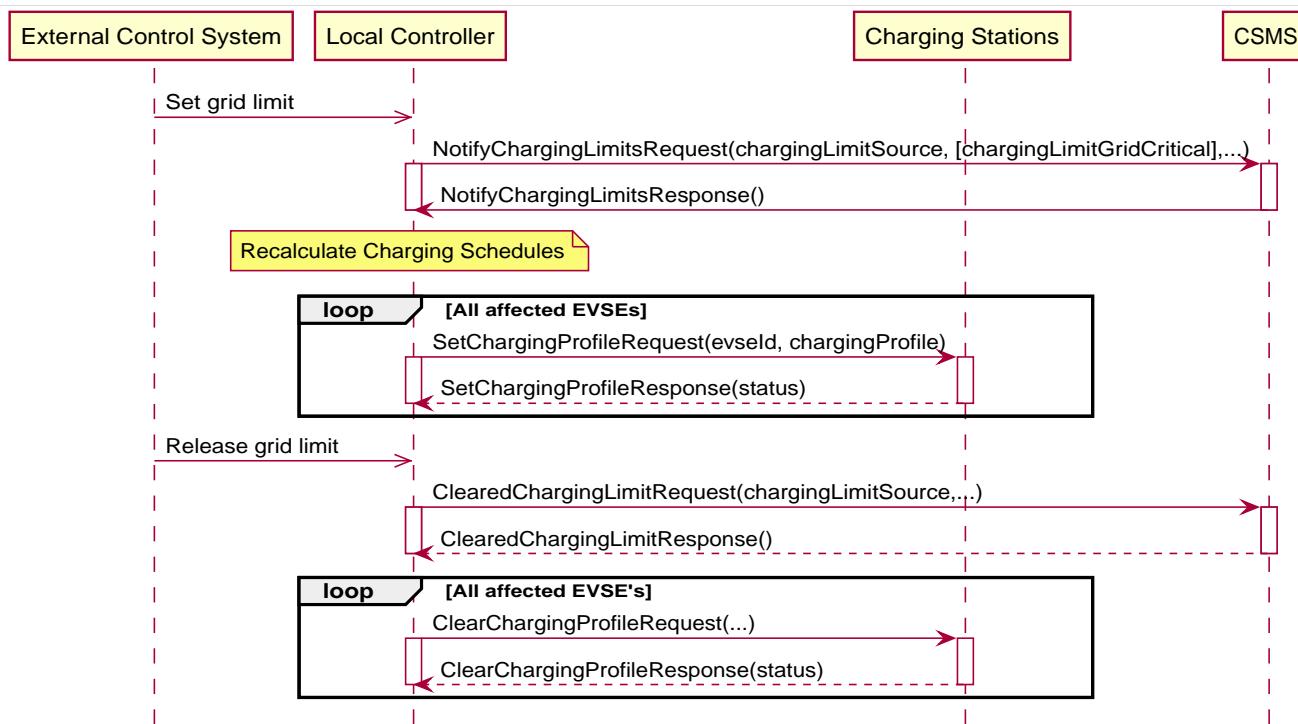


Figure 114. Sequence Diagram: External Charging Limit with Local Controller.

7	Error handling	n/a
8	Remark(s)	n/a

K14 - External Charging Limit with Local Controller - Requirements

Table 186. K14 - Requirements

ID	Precondition	Requirement definition
K14.FR.01	When an external charging limit/schedule is received	The total load of all Charging Stations SHALL NOT exceed this given limit.
K14.FR.02	K14.FR.01 AND Charging limit changed by more than: LimitChangeSignificance	The Local Controller SHALL inform the CSMS of the new charging limit/schedule imposed by the external system by sending a NotifyChargingLimitRequest .
K14.FR.03	When an external charging limit/schedule is released	The local controller SHALL notify the CSMS by sending a ClearedChargingLimitRequest .
K14.FR.04	K14.FR.03	The local controller SHALL clear the hard limit on Charging Stations by sending a ClearChargingProfileRequest message to the Charging Stations.
K14.FR.05	When the Local Controller receives an external charging limit/schedule	It SHALL send a SetChargingProfileRequest to all Charging Stations for which the charging profile has changed.
K14.FR.06	K14.FR.05	The Local Controller SHALL NOT set chargingProfilePurpose to ChargingStationExternalConstraints .

5.3. ISO 15118 based Smart Charging

K15 - Charging with load leveling based on High Level Communication

Table 187. K15 - Charging with load leveling based on High Level Communication

No.	Type	Description
1	Name	Charging with load leveling based on High Level Communication.
2	ID	K15
	Functional block	K. Smart Charging
	Reference	ISO15118-1 E1 AC Charging with load leveling based on High Level Communication, and E4 DC charging with load leveling based on High Level Communication.
3	Objectives	See ISO15118-1 , use case Objective E1, page 29.
4	Description	See ISO15118-1 , use case Description E1, page 29.
5	Actors	EV, Charging Station, CSMS.
6	Combined scenario description	<ol style="list-style-type: none"> 1. The EV sends a ChargeParameterDiscoveryReq message to the Charging Station. 2. The Charging Station sends a NotifyEVChargingNeedsRequest message to the CSMS. 3. The CSMS sends a NotifyEVChargingNeedsResponse message to the Charging Station. 4. The CSMS sends a SetChargingProfileRequest message to the Charging Station. 5. The Charging Station sends a SetChargingProfileResponse message to the CSMS. 6. The Charging Station responds to the EV with a ChargeParameterDiscoveryRes message to the EV. 7. The EV sends a PowerDeliveryReq message to the Charging Station with ChargeProgress=Start. This marks the point in time when the EVSE provides voltage to its output power outlet and the EV can start to recharge its battery. 8. The contactor is closed. 9. The transaction is updated with a TransactionEventRequest message. 10. A PowerdeliveryRes message is sent to the EV. 11. Optionally, the Charging Station sends a NotifyEVChargingScheduleRequest message to the CSMS.
7	Prerequisites	Both the Charging Station and the EV support ISO 15118.
8	Postcondition(s)	See ISO15118-1 , use case End conditions E1, page 29.

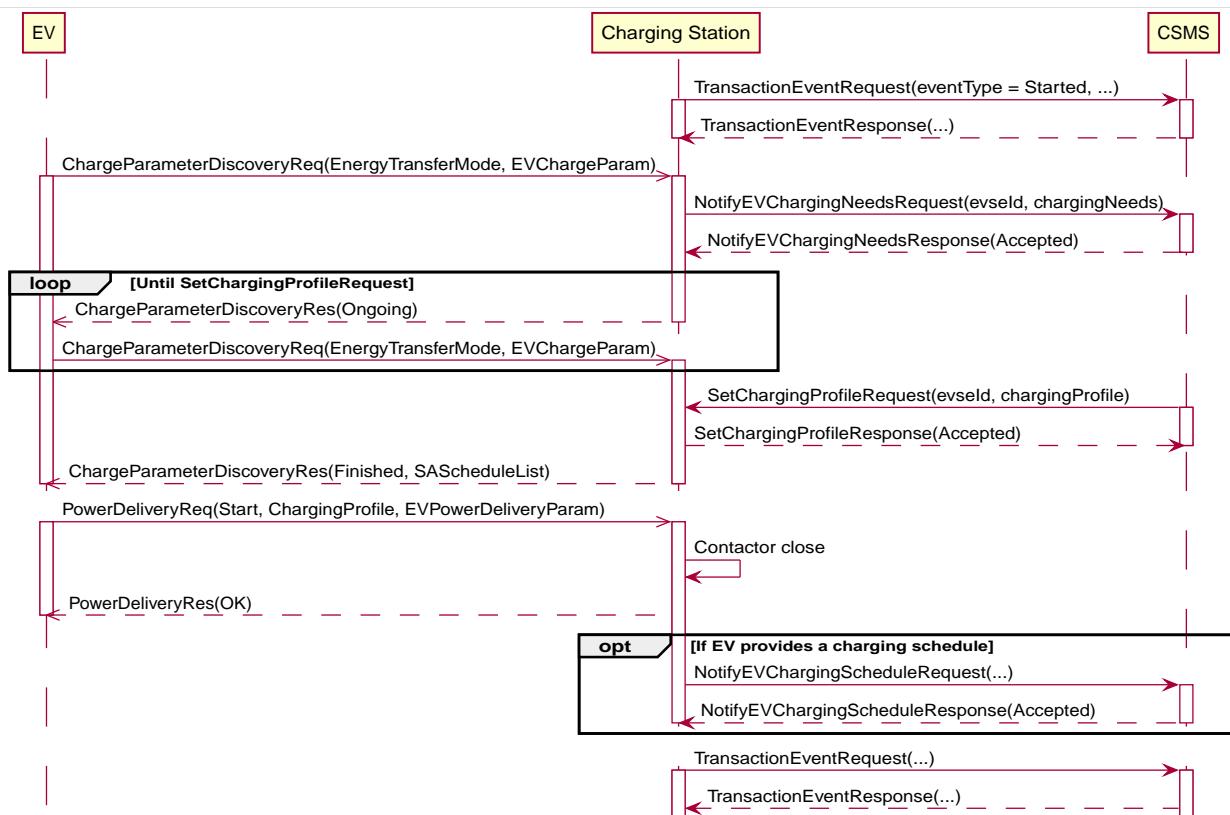


Figure 115. Sequence Diagram: Charging with load leveling based on High Level Communication

9	Error handling	The Charging Station needs to use the information from the SetChargingProfileRequest message to create the response to the ISO 15118 ChargeParameterDiscoveryReq towards the EV. This message has a timeout of 60 seconds, which means the SetChargingProfileRequest has to be sent well within 60 seconds after receiving the NotifyEVChargingNeedsRequest. If the Charging Station does not receive the SetChargingProfileRequest in time or when the NotifyEVChargingNeedsResponse has status = Processing, then the Charging Station will return a schedule in ChargeParameterDiscoverRes that matches the capabilities of the EVSE. When CSMS sends the SetChargingProfileRequest at a later time, then this will trigger a renegotiation according to use case K16 - Renegotiation initiated by CSMS.
10	Remark(s)	Signed SalesTariffs are currently not supported. If these are needed please use P01 - Data Transfer to the Charging Station to send these to the Charging Station.

K15 - Charging with load leveling based on High Level Communication - Requirements

Table 188. K15 - Requirements

ID	Precondition	Requirements	Note
K15.FR.01	When the Charging Station receives charging needs from the EV	The Charging Station SHALL send a NotifyEVChargingNeedsRequest to the CSMS.	
K15.FR.02	K15.FR.01	In response to a NotifyEVChargingNeedsRequest the CSMS SHALL send a NotifyEVChargingNeedsResponse .	
K15.FR.03	K15.FR.02	If the CSMS is able to provide a charging schedule, it SHALL indicate this by setting the status field in the NotifyEVChargingNeedsResponse to 'Accepted'.	

ID	Precondition	Requirements	Note
K15.FR.04	K15.FR.02	If the CSMS is not able to provide a charging schedule, it SHALL indicate this by setting the <i>status</i> field in the NotifyEVChargingNeedsResponse to 'Rejected'.	
K15.FR.05	K15.FR.02	If the CSMS is able to provide a charging schedule; but needs processing time, it SHALL indicate this by setting the <i>status</i> field in the NotifyEVChargingNeedsResponse to 'Processing'.	The Charging Station does not have to wait for the SetChargingProfileRequest . CSMS will send it later and trigger a renegotiation as per use case K16.
K15.FR.06		A NotifyEVChargingNeedsRequest SHALL contain either ACChargingParameters or DCChargingParameters.	
K15.FR.07	K15.FR.03 or K15.FR.05	The CSMS SHALL send a SetChargingProfileRequest with <i>chargingProfilePurpose</i> = TxProfile and a <i>transactionId</i> and at most three <i>chargingSchedule</i> and optional <i>salesTariff</i> elements, that each contain no more periods than specified by <i>maxScheduleTuples</i> in NotifyEVChargingNeedsRequest and by device model variable <code>SmartChargingCtrlr.PeriodsPerSchedule</code> .	The Charging Station will calculate the composite schedule(s) for the EVSE (taking into account a <code>ChargingStationMaxProfile</code> or <code>ChargingStationExternalConstraints</code> if present) and will convert that to the SAScheduleList format for ISO 15118.
K15.FR.08	K15.FR.01	The CSMS SHOULD send a SetChargingProfileRequest to the Charging Station within 60 seconds.	This is to satisfy the ISO 15118 ChargeParameterDiscoveryReq timeout.
K15.FR.09	K15.FR.07 AND EV returns a charging profile	Charging Station SHALL verify that provided charging profile is within boundaries of the ChargingSchedule from CSMS.	In ISO 15118 EV can sent its charging profile as part of PowerDeliveryReq.
K15.FR.10	K15.FR.09	Charging Station SHALL send the EV charging profile in a NotifyEVChargingScheduleRequest message to CSMS.	
K15.FR.11	K15.FR.10 AND EV charging profile is within limits of CSMS ChargingSchedule	CSMS responds with NotifyEVChargingScheduleResponse with status Accepted to Charging Station.	Note: Already checked by Charging Station, but CSMS does its own check.
K15.FR.12	K15.FR.10 AND EV charging profile is NOT within limits of CSMS ChargingSchedule	CSMS responds with NotifyEVChargingScheduleResponse with status Rejected to Charging Station.	
K15.FR.13	K15.FR.12	CSMS starts new renegotiation as per use case K16.	
K15.FR.14	K15.FR.11	The Charging Station SHOULD take the schedule from the NotifyEVChargingScheduleRequest into account when calculating the actual Composite schedule.	
K15.FR.15	K15.FR.03 AND Charging Station is offline	The Charging Station SHALL use the <i>TxDefaultProfile</i> (if present) and generate a charging schedule within the limits of its composite schedule.	
K15.FR.16	K15.FR.07	It is RECOMMENDED to configure the Charging Station, such that a TransactionEvent with <i>idToken</i> has been sent prior to the NotifyEVChargingNeedsRequest Message, so that CSMS can take the user into account when creating a charging schedule.	

ID	Precondition	Requirements	Note
K15.FR.17	When Charging Station receives a SetChargingProfileRequest immediately after the transaction has started and before it has sent the NotifyEVChargingNeedsRequest to CSMS	The Charging Station SHOULD respond with SetChargingProfileResponse with status = Rejected and a statusInfo with reasonCode = InvalidMessageSequence.	CSMS sent profile too early. It does not harm if CS accepts the charging profile instead of rejecting it, as long as it sends a charging profile again when it receives the NotifyEVChargingNeedsRequest .
K15.FR.18	K15.FR.03 OR K15.FR.05	CSMS IS RECOMMENDED to use only one chargingSchedule in a SetChargingProfileRequest .	This ensures that there is no doubt about which schedule the EV will follow, even when no NotifyEVChargingScheduleRequest is received.
K15.FR.19	K15.FR.07 AND EV does not return a charging profile	Charging Station IS RECOMMENDED to return an EV charging profile as a chargingSchedule in a NotifyEVChargingScheduleRequest message to CSMS that matches the schedule that was selected by the EV (i.e. chargingSchedule.id = SAScheduleTupleId)	In ISO 15118-2 the EV charging profile and the selected schedule are returned as ChargingProfile and SAScheduleTupleId in PowerDeliveryReq .

K16 - Renegotiation initiated by CSMS

Table 189. K16 - Renegotiation initiated by CSMS

No.	Type	Description
1	Name	Renegotiation initiated by CSMS.
2	ID	K16
	Functional block	K. Smart Charging
3	Objectives	To control the charging power or current of a Charging Station
4	Description	<p>The CSMS sends a SetChargingProfileRequest to the Charging Station to influence the power or current drawn by the EV. The CSMS calculates a ChargingSchedule to stay within limits which MAY be imposed by an external system.</p> <p>Note: Description of actions between EV and Charging Station is informative only and not mandated by OCPP.</p>
	Actors	EV, Charging Station, CSMS
	Scenario description	<p>1 CSMS sends a SetChargingProfileRequest to the Charging Station.</p> <p>2 Charging Station responds with a SetChargingProfileResponse to the CSMS.</p> <p>3 When EV sends the next CurrentDemandReq (for DC) or ChargingStatusReq (for AC), the Charging Station will respond with evseNotification = ReNegotiation.</p> <p>4 EV sends a PowerDeliveryReq with chargeProgress = ReNegotiate to confirm this.</p> <p>5 Charging Station responds with a PowerDeliveryRes.</p> <p>6 EV sends a ChargeParameterDiscoveryReq.</p> <p>7 Charging Station responds with a ChargeParameterDiscoveryRes with an SAScheduleList that contains the ChargingSchedule data from the SetChargingProfileRequest.</p> <p>8 EV sends a PowerDeliveryReq with chargeProgress = Start (with an optional charging profile) to confirm this.</p> <p>9 Charging Station responds with PowerDeliveryRes and, if charging was suspended at start of the renegotiation, will resume power delivery.</p> <p>10 If EV provided a charging profile in the previous step, then Charging Station will send a NotifyEVChargingScheduleRequest to the CSMS.</p>
5	Prerequisites	Charging session started according to use case K15.
6	Postcondition(s)	Charging session uses the new charging profile.

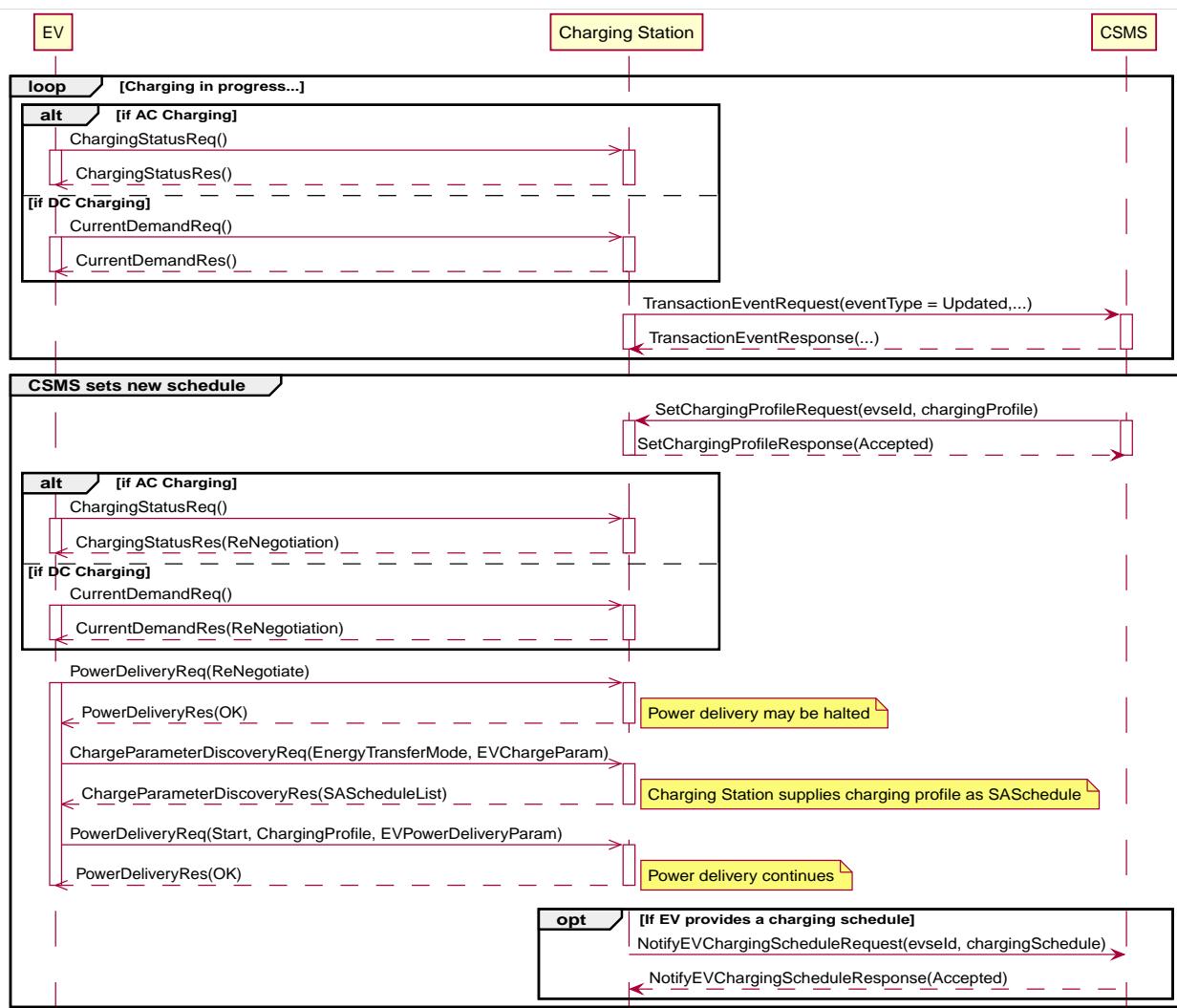


Figure 116. Renegotiation initiated by CSMS

7	Remark(s)	Signed SalesTariffs are currently not supported. If these are needed please use P01 - Data Transfer to the Charging Station to send these to the Charging Station.
---	-----------	--

K16 - Renegotiation initiated by CSMS - Requirements

ID	Precondition	Requirements	NOTE
K16.FR.01	CSMS sends a new SetChargingProfileRequest	Charging Station SHALL respond with a SetChargingProfileResponse with status = Accepted.	
K16.FR.02	K16.FR.01	Charging Station SHALL initiate schedule renegotiation with EV.	In ISO 15118 this is done by replying with EVSENotification=ReNegotiation to a CurrentDemandReq (for DC) or ChargingStatusReq (for AC) message.
K16.FR.03	K16.FR.02	Charging Station SHALL provide the ChargingSchedule data to the EV.	In ISO 15118 this is done in the ChargeParameterDiscoverRes message.
K16.FR.04	EV returns a charging profile	Charging Station SHALL verify that provided charging profile is within boundaries of the ChargingSchedule from CSMS.	In ISO 15118 EV may provide this as part of the PowerDeliveryReq message.
K16.FR.05	K16.FR.04	Charging Station SHALL send the EV charging profile in a NotifyEVChargingScheduleRequest message to CSMS.	
K16.FR.06	K16.FR.05 AND EV charging profile is within limits of CSMS ChargingSchedule	CSMS responds with NotifyEVChargingScheduleResponse with status Accepted to Charging Station.	Note: Already checked by Charging Station, but CSMS does its own check.

ID	Precondition	Requirements	NOTE
K16.FR.07	K16.FR.05 AND EV charging profile is NOT within limits of CSMS ChargingSchedule	CSMS responds with NotifyEVChargingScheduleResponse with status Rejected to Charging Station.	
K16.FR.08	K16.FR.07	CSMS starts new renegotiation as per use case K16.	
K16.FR.09	When the Charging Station receives charging needs from the EV	The Charging Station SHOULD NOT send a NotifyEVChargingNeedsRequest to the CSMS.	CSMS initiated the renegotiation and has just sent a new charging profile, based on the initial charging needs from EV, energy already consumed by EV and whatever information has caused CSMS to update the charging profile. In ISO 15118 charging needs are sent via ChargeParameter-DiscoveryReq.
K16.FR.10	K16.FR.04	The Charging Station SHOULD take the schedule from the NotifyEVChargingScheduleRequest into account when calculating the actual Composite schedule.	
K16.FR.11	K16.FR.02 AND current or power in new charging schedule is lower than actual current or power	The Charging Station SHALL request EV to lower current or power to a value matching the new charging schedule at the first possible opportunity.	In ISO 15118 this can be communicated in CurrentDemandRes (for DC) or ChargingStatusRes (for AC).
K16.FR.12	K16.FR.09 AND Charging Station sends a NotifyEVChargingNeed sRequest	The CSMS SHALL send a SetChargingProfileRequest .	This situation is not desirable, because charging profile will likely be the same as in K16.FR.01, but this is added for robustness when Charging Station is not adhering to K16.FR.09.
K16.FR.13	EV does not return a charging profile	Charging Station IS RECOMMENDED to return an EV charging profile as a chargingSchedule in a NotifyEVChargingScheduleRequest message to CSMS that matches the schedule that was selected by the EV (i.e. chargingSchedule.id = SAScheduleTupleId)	In ISO 15118-2 the EV charging profile and the selected schedule are returned as <i>ChargingProfile</i> and <i>SAScheduleTupleId</i> in PowerDeliveryReq.

K17 - Renegotiation initiated by EV

Table 190. K17 - Renegotiation initiated by EV

No.	Type	Description
1	Name	Renegotiation initiated by EV.
2	ID	K16
	Functional block	K. Smart Charging
3	Objectives	To let an EV request a new charging schedule.
4	Description	<p>The EV signals the Charging Station that it wants to renegotiate and it provides new charging needs, which the Charging Station sends to the CSMS. Based on this and other parameters, the CSMS calculates a new charging schedule and sends it via SetChargingProfileRequest to Charging Station, which communicates it to the EV.</p> <p>Note: Description of actions between EV and Charging Station is informative only and not mandated by OCPP.</p>
	Actors	EV, Charging Station, CSMS

No.	Type	Description
	Scenario description	<p>1 When EV sends a ChargeParameterDiscoveryReq with with charging needs parameters, then Charging Station sends this information in a NotifyEVChargingNeedsRequest to CSMS.</p> <p>2 CSMS responds with NotifyEVChargingNeedsResponse to Charging Station.</p> <p>3 CSMS calculates new charging schedule, that tries to accomodate the EV charging needs and still fits within the schedule boundaries imposed by other parameters.</p> <p>4 CSMS sends a SetChargingProfileRequest with the new schedule to the Charging Station.</p> <p>5 Charging Station responds with SetChargingProfileResponse with status Accepted.</p> <p>6 Charging Station sends new charging schedule to EV in a ChargeParameterDiscoveryRes message.</p> <p>7 EV sends a PowerDeliveryReq with chargeProgress = Start (with an optional charging profile) to confirm this.</p> <p>8 Charging Station responds with PowerDeliveryRes and, if charging was suspended at start of the renegotiation, will resume power delivery.</p> <p>9 If EV provided a charging profile in the previous step, then Charging Station will send a NotifyEVChargingScheduleRequest to the CSMS.</p>
5	Prerequisites	Charging session started according to use case K15.
6	Postcondition(s)	Charging session uses the new charging profile.

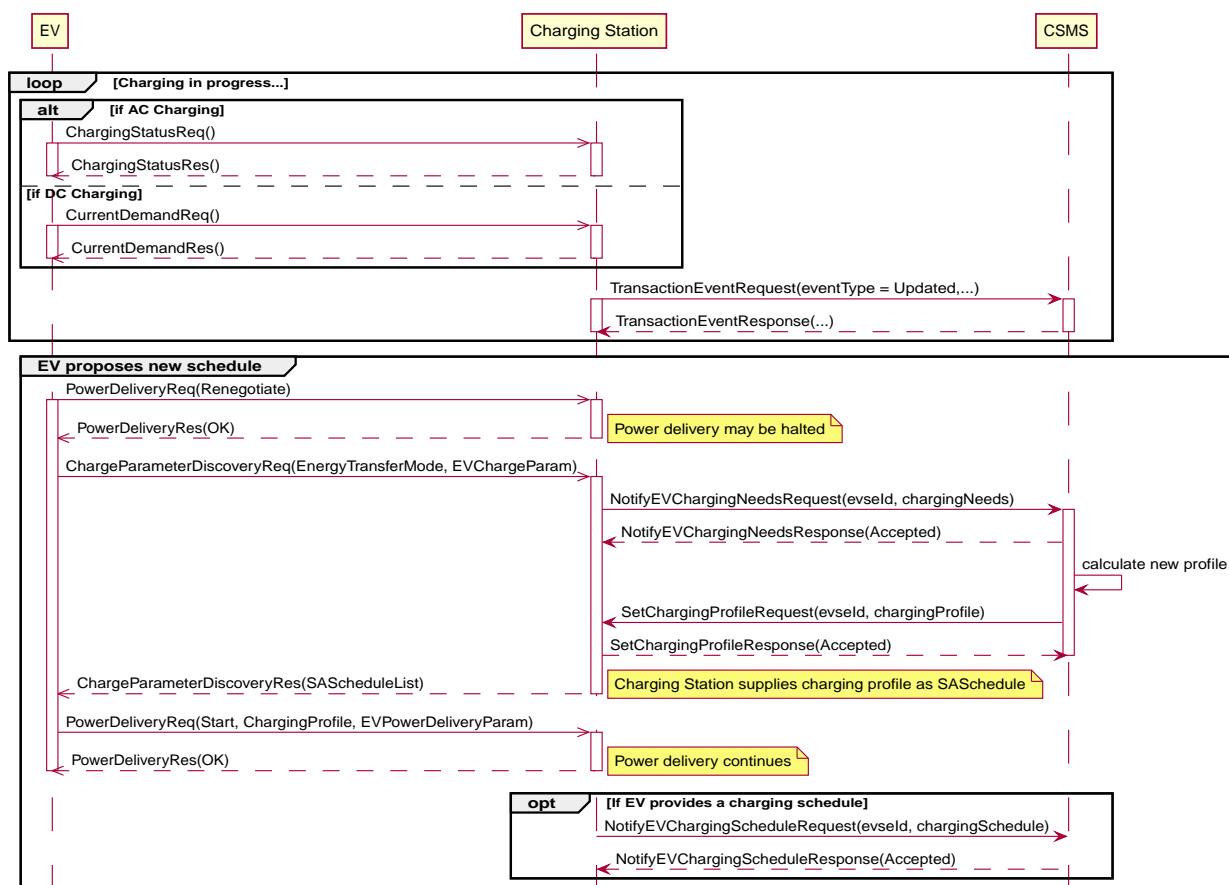


Figure 117. Renegotiation initiated by EV

7	Remark(s)	Signed SalesTariffs are currently not supported. If these are needed please use P01 - Data Transfer to the Charging Station to send these to the Charging Station.
---	-----------	--

K17 - Renegotiation initiated by EV - Requirements

Table 191. K17 - Requirements

ID	Precondition	Requirements	Note
K17.FR.01	EV triggers a renegotiation and sends new charging needs	The Charging Station SHALL send a NotifyEVChargingNeedsRequest to the CSMS.	

ID	Precondition	Requirements	Note
K17.FR.02	K17.FR.01	In response to a NotifyEVChargingNeedsRequest the CSMS SHALL send a NotifyEVChargingNeedsResponse .	
K17.FR.03	K17.FR.02	If the CSMS is able to provide a charging schedule, it SHALL indicate this by setting the <i>status</i> field in the NotifyEVChargingNeedsResponse to 'Accepted'.	
K17.FR.04	K17.FR.02	If the CSMS is not able to provide a charging schedule, it SHALL indicate this by setting the <i>status</i> field in the NotifyEVChargingNeedsResponse to 'Rejected'.	
K17.FR.05	K17.FR.02	If the CSMS is able to provide a charging schedule; but needs processing time, it SHALL indicate this by setting the <i>status</i> field in the NotifyEVChargingNeedsResponse to 'Processing'.	
K17.FR.06		A NotifyEVChargingNeedsRequest SHALL contain either ACChargingParameters or DCChargingParameters.	
K17.FR.07	K17.FR.03 or K17.FR.05	The CSMS SHALL send a SetChargingProfileRequest with <i>chargingProfilePurpose</i> = TxProfile and at most three <i>chargingSchedule</i> and optional <i>salesTariff</i> elements, that each contain no more periods than specified by <i>maxScheduleTuples</i> in NotifyEVChargingNeedsRequest and by device model variable SmartChargingCtrlr.PeriodsPerSchedule.	
K17.FR.08	K17.FR.01	The CSMS SHOULD send a SetChargingProfileRequest to the Charging Station within 60 seconds.	This is to satisfy the ISO 15118 ChargeParameterDiscoveryReq timeout.
K17.FR.09	K17.FR.07 AND EV returns a charging profile	Charging Station SHALL verify that provided charging profile is within boundaries of the ChargingSchedule from CSMS.	In ISO 15118 EV can sent its charging profile as part of PowerDeliveryReq.
K17.FR.10	K17.FR.09	Charging Station SHALL send the EV charging profile in a NotifyEVChargingScheduleRequest message to CSMS.	
K17.FR.11	K17.FR.10 AND EV charging profile is within limits of CSMS ChargingSchedule	CSMS responds with NotifyEVChargingScheduleResponse with <i>status</i> Accepted to Charging Station.	Note: Already checked by Charging Station, but CSMS does its own check.
K17.FR.12	K17.FR.10 AND EV charging profile is NOT within limits of CSMS ChargingSchedule	CSMS responds with NotifyEVChargingScheduleResponse with <i>status</i> Rejected to Charging Station.	
K17.FR.13	K17.FR.12	CSMS starts new renegotiation as per use case K16.	
K17.FR.14	K17.FR.11	The Charging Station SHOULD take the schedule from the NotifyEVChargingScheduleRequest into account when calculating the actual Composite schedule.	

ID	Precondition	Requirements	Note
K17.FR.15	K17.FR.01 AND Charging Station is offline	The Charging Station SHALL use the TxDefaultProfile (if present) and generate a charging schedule within the limits of its composite schedule.	
K17.FR.16	K17.FR.07 EV does not return a charging profile	Charging Station IS RECOMMENDED to return an EV charging profile as a chargingSchedule in a NotifyEVChargingScheduleRequest message to CSMS that matches the schedule that was selected by the EV (i.e. chargingSchedule.id = SAScheduleTupleId)	In ISO 15118-2 the EV charging profile and the selected schedule are returned as <i>ChargingProfile</i> and <i>SAScheduleTupleId</i> in <i>PowerDeliveryReq</i> .

L. FirmwareManagement

1. Introduction

This Functional Block describes the functionality that enables a CSO to update the firmware of a Charging Station.

When a Charging Station needs to be updated with new firmware, the CSMS informs the Charging Station of the time at which the Charging Station can start downloading the new firmware. The Charging Station SHALL notify the CSMS after each step as it downloads and installs the new firmware.

2. Use cases & Requirements

L01 - Secure Firmware Update

Table 192. L01 - Secure Firmware Update

No.	Type	Description
1	Name	Secure Firmware Update
2	ID	L01
	Functional block	L. Firmware Management
3	Objective(s)	Download and install a Secure firmware update.
4	Description	Illustrate how a Charging Station processes a Secure firmware update.
	Actors	CSMS, Charging Station, Charging Station Manufacturer
	Scenario description	<p>1. The CSMS sends an UpdateFirmwareRequest message that contains the location of the firmware, the time after which it should be retrieved, and information on how many times the Charging Station should retry downloading the firmware.</p> <p>2. The Charging Station verifies the validity of the certificate against the Manufacturer root certificate.</p> <p>3. If the certificate is valid, the Charging Station starts downloading the firmware, and sends a FirmwareStatusNotificationRequest with status Downloading. If the certificate is not valid or could not be verified, the Charging Station aborts the firmware update process and sends a UpdateFirmwareResponse with status InvalidCertificate and a SecurityEventNotificationRequest with the security event InvalidFirmwareSigningCertificate (See part 2 appendices for the full list of security events).</p> <p>4. If the Firmware successfully downloaded, the Charging Station sends a FirmwareStatusNotificationRequest with status Downloaded. Otherwise, it sends a FirmwareStatusNotificationRequest with status DownloadFailed.</p> <p>5. If the verification is successful, the Charging Station sends a FirmwareStatusNotificationRequest with status Installing. If the verification of the firmware fails or if a signature is missing entirely, the Charging Station sends a FirmwareStatusNotificationRequest with status InvalidSignature and a SecurityEventNotificationRequest with the security event InvalidFirmwareSignature (See part 2 appendices for the full list of security events).</p> <p>6. If the installation is successful, the Charging Station sends a FirmwareStatusNotificationRequest with status Installed. Otherwise, it sends a FirmwareStatusNotificationRequest with status InstallationFailed.</p>
	Alternative scenario(s)	L02 - Non-Secure Firmware Update
5	Prerequisite(s)	The Charging Station Manufacturer provided a firmware update.
6	Postcondition(s)	<p>Successful postcondition: The firmware is updated and the Charging Station is in Installed status.</p> <p>Failure postconditions: The certificate is not valid or could not be verified and the Charging Station is in InvalidCertificate status. Downloading the firmware failed and the Charging Station is in DownloadFailed status. The verification of the firmware's digital signature failed and the Charging Station is in InvalidSignature status. The installation of the firmware is not successful and the Charging Station is in InstallationFailed status.</p>

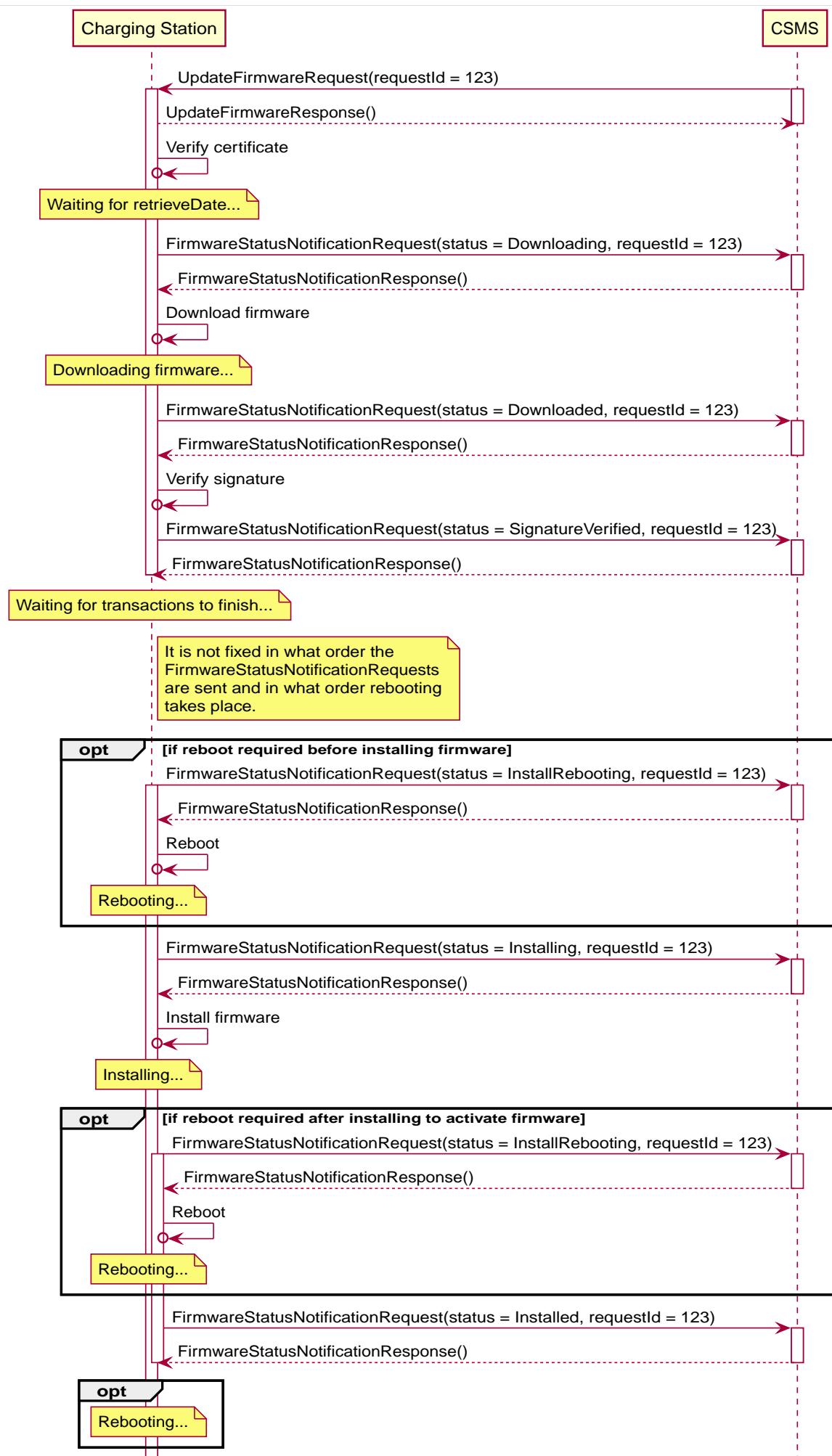


Figure 118. Sequence diagram secure firmware upgrade (happy flow)

7	Error handling	n/a
8	Remark(s)	<p>As an example in this use case the requestId = 123, but this could be any value.</p> <p>Measures SHOULD be taken to secure the firmware when it is stored on a server or workstation.</p> <p>The Charging Station has a required Configuration Variable that reports which file transfer protocols it supports: FileTransferProtocols</p> <p>When migrating to a new version of OCPP it is RECOMMENDED to install a fallback NetworkConnectionProfile with the new configuration.</p> <p>The requirements for the Firmware Signing Certificate are described in the: Certificate Properties section.</p> <p>The manufacturer SHALL NOT use intermediate certificates for the firmware signing certificate in the Charging Station.</p> <p>FTP needs to be able to use Passive FTP, to be able to transverse over as much different typologies as possible.</p>

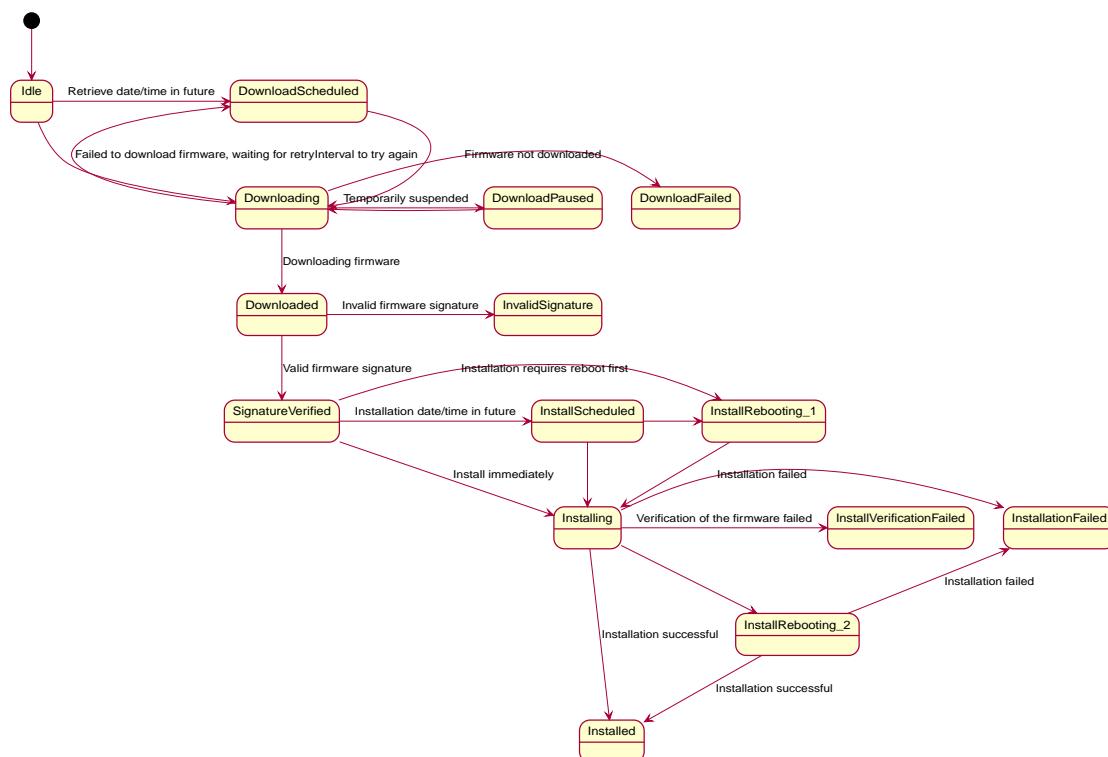


Figure 119. Firmware update process

L01 - Secure Firmware Update - Requirements

Table 193. L01 - Requirements

ID	Precondition	Requirement definition	Note
L01.FR.01	Whenever the Charging Station enters a new state in the firmware update process.	The Charging Station SHALL send a FirmwareStatusNotificationRequest message to the CSMS with this new status. What reason to use is described in the description of FirmwareStatusEnumType .	
L01.FR.02	When the Charging Station enters the Invalid Certificate state in the firmware process.	The Charging Station SHALL send a SecurityEventNotificationRequest message to the CSMS with the security event InvalidFirmwareSigningCertificate (See part 2 appendices for the full list of security events).	

ID	Precondition	Requirement definition	Note
L01.FR.03	When the Charging Station enters the Invalid Signature state.	The Charging Station SHALL send a SecurityEventNotificationRequest message to the CSMS with the security event InvalidFirmwareSignature (See part 2 appendices for the full list of security events).	
L01.FR.04	When the Charging Station has successfully downloaded the new firmware	The signature SHALL be validated, by calculating the signature over the entire firmware file using the RSA-PSS or EC Schnorr algorithm for signing, and the SHA256 algorithm for calculating hash values.	
L01.FR.05	L01.FR.04 AND (<i>installDateTime</i> is not set OR current time >= <i>installDateTime</i>)	The Charging Station SHALL install the new firmware as soon as it is able to.	
L01.FR.06	L01.FR.05 AND The Charging Station has ongoing transactions AND When it is not possible to continue charging during installation of firmware	The Charging Station SHALL wait until all transactions have ended, before commencing installation.	
L01.FR.07	L01.FR.06 AND configuration variable AllowNewSessionsPendingFirmwareUpdate is <i>false</i> or does not exist	The Charging Station SHALL set all connectors that are not in use to UNAVAILABLE while the Charging Station waits for the ongoing transactions to end. Until the firmware is installed, any connector that becomes available SHALL be set to UNAVAILABLE.	
L01.FR.08		It is RECOMMENDED that the firmware is sent encrypted to the Charging Station. This can either be done by using a secure protocol (such as HTTPS, SFTP, or FTPS) to send the firmware, or by encrypting the firmware itself before sending it.	
L01.FR.09		Firmware updates SHALL be digitally protected to ensure authenticity and to provide proof of origin.	This protection is achieved by applying a digital signature over the hash value of the firmware image. Ideally, this signature is already computed by the manufacturer. This way proof of origin of the firmware image can be tracked back to the original author of the firmware.
L01.FR.10		Every FirmwareStatusNotificationRequest sent for a firmware update SHALL contain the same requestId as the UpdateFirmwareRequest that started this firmware update.	
L01.FR.11		For security purposes the CSMS SHALL include the Firmware Signing certificate (see Keys used in OCPP) in the UpdateFirmwareRequest .	
L01.FR.12		For verifying the certificate (see Certificate Hierarchy) use the rules for X.509 certificates [19]. The Charging Station MUST verify the file's digital signature using the Firmware Signing certificate.	
L01.FR.13	When the Charging Station does not start downloading firmware, because it is busy charging or because <i>retrieveDateTime</i> is in the future	The Charging Station SHALL send a FirmwareStatusNotificationRequest with status DownloadScheduled .	
L01.FR.14	When the Charging Station enters the Download Paused state.	The Charging Station SHALL send a FirmwareStatusNotificationRequest with status DownloadPaused .	For example when the Charging Station has tasks with higher priorities.

ID	Precondition	Requirement definition	Note
L01.FR.15	When a Charging Station needs to reboot before installing the downloaded firmware.	The Charging Station SHALL send a FirmwareStatusNotificationRequest with status InstallRebooting , before rebooting.	
L01.FR.16	L01.FR.04 AND When <i>installDateTime</i> is set to a time in the future	The Charging Station SHALL send a FirmwareStatusNotificationRequest with status InstallScheduled and install the firmware at the specified installation time.	
L01.FR.20		The field <i>requestId</i> in FirmwareStatusNotificationRequest is mandatory, unless <i>status</i> = Idle .	
L01.FR.21	When the Charging Station receives an UpdateFirmwareRequest	The Charging Station SHALL validate the certificate before accepting the message.	
L01.FR.22	L01.FR.21 AND the certificate is invalid	The Charging Station SHALL respond with UpdateFirmwareResponse with status InvalidCertificate .	
L01.FR.23	When the Charging Station needs to reboot during a firmware update AND the bootloader is unable to send OCPP messages	The Charging Station MAY omit the FirmwareStatusNotificationRequest message with status Installing .	
L01.FR.24	When a Charging Station is installing new Firmware OR is going to install new Firmware, but has received an UpdateFirmware command to install it at a later time AND the Charging Station receives a new UpdateFirmwareRequest	The Charging Station SHOULD cancel the ongoing firmware update AND respond with status AcceptedCanceled .	The Charging Station SHOULD NOT first check if the new firmware file exists, this way the CSMS will be able to cancel an ongoing firmware update without starting a new one.
L01.FR.25	Charging Station receives a TriggerMessageRequest for FirmwareStatusNotification AND last sent FirmwareStatusNotificationRequest had <i>status</i> = Installed	Charging Station SHALL return a FirmwareStatusNotificationRequest with <i>status</i> = Idle .	
L01.FR.26	Charging Station receives a TriggerMessageRequest for FirmwareStatusNotification AND last sent FirmwareStatusNotificationRequest had NOT <i>status</i> Installed	Charging Station SHALL return a FirmwareStatusNotificationRequest with the last sent <i>status</i> .	
L01.FR.27	L01.FR.24 AND the Charging Station is unable to cancel the firmware installation	The Charging Station MAY respond with <i>status</i> = Rejected .	
L01.FR.28	When the Charging Station has successfully installed the new firmware	The Charging Station SHALL send a FirmwareStatusNotificationRequest with status Installed AND The Charging Station SHOULD have activated the new firmware already or do so immediately.	Activating the new firmware MAY involve an automatic reboot, but not necessarily so.
L01.FR.29	If the verification of the new firmware (e.g. using a checksum or some other means) fails	The Charging Station SHALL send a FirmwareStatusNotificationRequest with status InstallVerificationFailed	
L01.FR.30	When the Charging Station has failed all retry attempts to download the firmware.	The Charging Station SHALL send a FirmwareStatusNotificationRequest with status DownloadFailed .	A Charging Station MAY send a new Downloading status upon each retry attempt.
L01.FR.31	L01.FR.28	The Charging Station SHALL send a SecurityEventNotificationRequest message with <i>type</i> = "FirmwareUpdated".	

ID	Precondition	Requirement definition	Note
L01.FR.32	When a Charging Station needs to reboot before activating the downloaded firmware	The Charging Station MAY send a FirmwareStatusNotificationRequest with status InstallRebooting , before rebooting.	

L02 - Non-Secure Firmware Update

Table 194. L02 - Non-Secure Firmware Update

No.	Type	Description
1	Name	Non-Secure Firmware Update
2	ID	L02
	Functional block	L. Firmware Management
3	Objective(s)	Download and install a Non-Secure firmware update.
4	Description	Illustrate how a Charging Station processes a Non-Secure firmware update.
	Actors	CSMS, Charging Station
	Scenario description	<p>1. The CSMS sends an UpdateFirmwareRequest message that contains the location of the firmware, the time after which it should be retrieved, and information on how many times the Charging Station should retry downloading the firmware.</p> <p>2. The Charging station responds with an UpdateFirmwareResponse.</p> <p>3. The Charging station sends a FirmwareStatusNotificationRequest with status <i>Downloading</i>.</p> <p>4. The CSMS responds with a FirmwareStatusNotificationResponse.</p> <p>5. The Charging station sends a FirmwareStatusNotificationRequest with status <i>Downloaded</i>.</p> <p>6. The CSMS responds with a FirmwareStatusNotificationResponse.</p> <p>7. The Charging station sends a FirmwareStatusNotificationRequest with status <i>Installing</i>.</p> <p>8. The CSMS responds with a FirmwareStatusNotificationResponse.</p> <p>9. The Charging station sends a FirmwareStatusNotificationRequest with status <i>Installed</i>.</p> <p>10. The CSMS responds with a FirmwareStatusNotificationResponse.</p>
	Alternative scenario(s)	L01 - Secure Firmware Update
5	Prerequisite(s)	The Charging Station Manufacturer provided a firmware update.
6	Postcondition(s)	Successful postcondition: Firmware update was successfully installed. Failure postcondition: Firmware update failed.

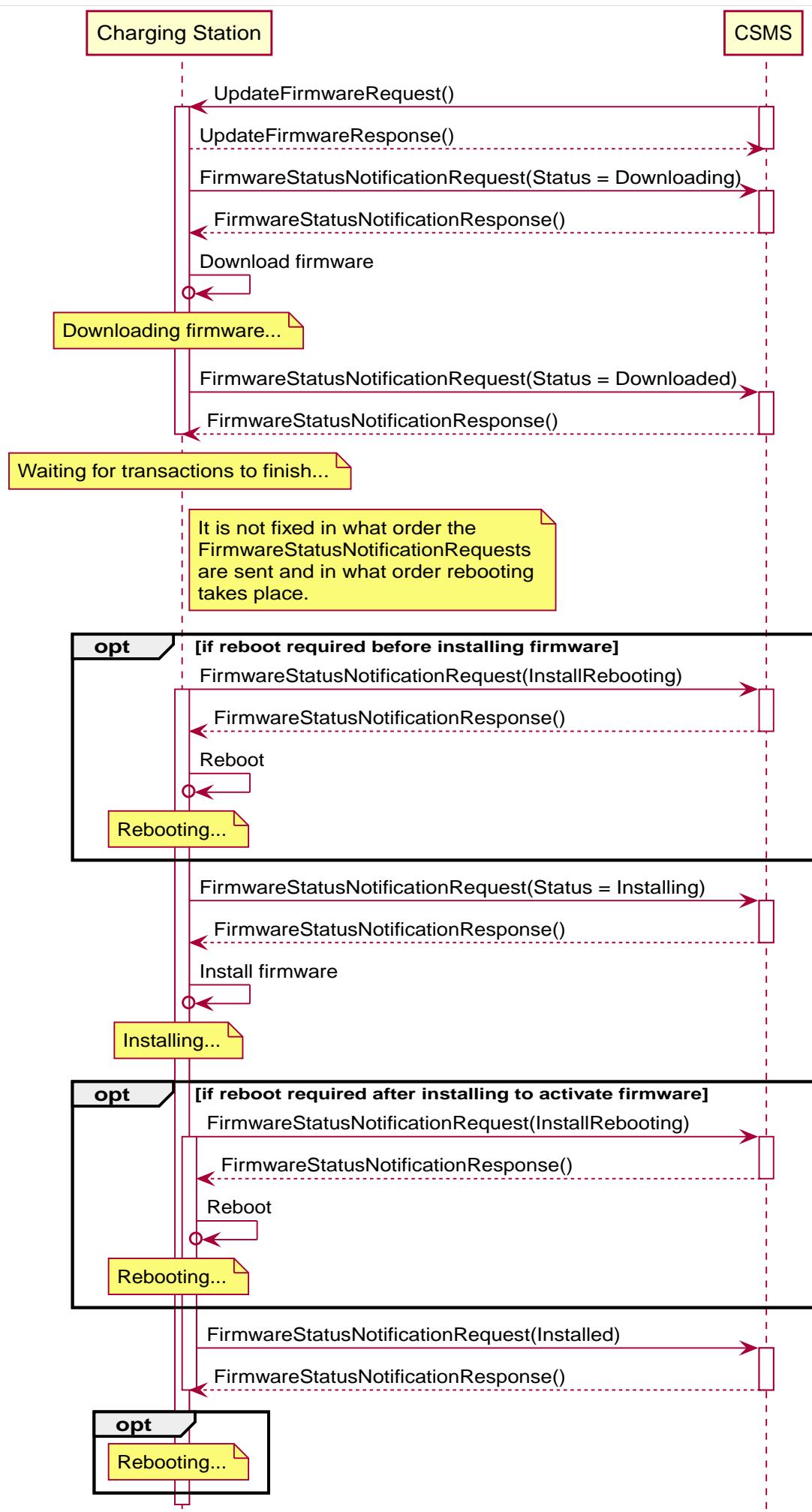


Figure 120. Sequence diagram Non-Secure firmware upgrade

7	Error handling	n/a
8	Remark(s)	<p>Measures SHOULD be taken to secure the firmware when it is stored on a server or workstation.</p> <p>When migrating to a new version of OCPP it is RECOMMENDED to install a fallback NetworkConnectionProfile with the new configuration.</p> <p>FTP needs to be able to use Passive FTP, to be able to transverse over as much different typologies as possible.</p>

L02 - Non-Secure Firmware Update - Requirements

Table 195. L02 - Requirements

ID	Precondition	Requirement definition	Note
L02.FR.01	Whenever the Charging Station enters a new status in the firmware update process.	The Charging Station SHALL send a FirmwareStatusNotificationRequest message to the CSMS with this new status.	
L02.FR.02	When the Charging Station has successfully downloaded the new firmware AND (<i>installDateTime</i> is not set OR current time >= <i>installDateTime</i>)	The Charging Station SHALL install the new firmware as soon as it is able to.	
L02.FR.03	L02.FR.02 AND The Charging Station has ongoing transactions AND When it is not possible to continue charging during installation of firmware	The Charging Station SHALL wait until all transactions have ended, before commencing installation.	
L02.FR.04	L02.FR.03 AND configuration variable AllowNewSessionsPendingFirmwareUpdate is false or does not exist	The Charging Station SHALL set all connectors that are not in use to UNAVAILABLE while the Charging Station waits for the ongoing transactions to end. Until the firmware is installed, any connector that becomes available SHALL be set to UNAVAILABLE.	
L02.FR.05		It is RECOMMENDED that the firmware is sent encrypted to the Charging Station. This can either be done by using a secure protocol (such as HTTPS, SFTP, or FTPS) to send the firmware, or by encrypting the firmware itself before sending it.	
L02.FR.06		Every FirmwareStatusNotificationRequest sent for a firmware update SHALL contain the same <i>requestId</i> as the UpdateFirmwareRequest that started this firmware update.	
L02.FR.07	When the Charging Station does not start downloading firmware, because it is busy charging or because <i>retrieveDateTime</i> is in the future	The Charging Station SHALL send a FirmwareStatusNotificationRequest with status DownloadScheduled .	
L02.FR.08	When the Charging Station enters the Download Paused state.	The Charging Station SHALL send a FirmwareStatusNotificationRequest with status DownloadPaused .	For example when the Charging Station has tasks with higher priorities.
L02.FR.09	When a Charging Station needs to reboot before installing the downloaded firmware.	The Charging Station SHALL send a FirmwareStatusNotificationRequest with status InstallRebooting , before rebooting.	
L02.FR.10	When the Charging Station has successfully downloaded the new firmware AND <i>installDateTime</i> is set to time in the future	The Charging Station SHALL send a FirmwareStatusNotificationRequest with status InstallScheduled and install the firmware at the specified installation time.	

ID	Precondition	Requirement definition	Note
L02.FR.14		The field <code>requestId</code> in <code>FirmwareStatusNotificationRequest</code> is mandatory, unless <code>status = Idle</code> .	
L02.FR.15	When a Charging Station is installing new Firmware OR is going to install new Firmware, but has received an <code>UpdateFirmware</code> command to install it at a later time AND the Charging Station receives a new <code>UpdateFirmwareRequest</code>	The Charging Station SHOULD cancel the ongoing firmware update AND respond with status <code>AcceptedCanceled</code> .	The Charging Station SHOULD NOT first check if the new firmware file exists, this way the CSMS will be able to cancel an ongoing firmware update without starting a new one.
L02.FR.16	Charging Station receives a <code>TriggerMessageRequest</code> for <code>FirmwareStatusNotificationRequest</code> AND last sent <code>FirmwareStatusNotificationRequest</code> had <code>status = Installed</code>	Charging Station SHALL return a <code>FirmwareStatusNotificationRequest</code> with <code>status = Idle</code> .	
L02.FR.17	Charging Station receives a <code>TriggerMessageRequest</code> for <code>FirmwareStatusNotificationRequest</code> AND last sent <code>FirmwareStatusNotificationRequest</code> had NOT <code>status Installed</code>	Charging Station SHALL return a <code>FirmwareStatusNotificationRequest</code> with the last sent <code>status</code> .	
L02.FR.18	L02.FR.15 AND the Charging Station is unable to cancel the firmware installation	The Charging Station MAY respond with <code>status = Rejected</code> .	
L02.FR.19	When the Charging Station has failed all retry attempts to download the firmware.	The Charging Station SHALL send a <code>FirmwareStatusNotificationRequest</code> with status <code>DownloadFailed</code> .	A Charging Station MAY send a new <code>Downloading</code> status upon each retry attempt.
L02.FR.20	When the Charging Station has successfully installed and activated the new firmware	The Charging Station SHALL send a <code>FirmwareStatusNotificationRequest</code> with status <code>Installed</code> .	Activation of the new firmware may involve a reboot.
L02.FR.21	When the Charging Station has successfully installed the new firmware AND the Charging Station needs to reboot before activating the new firmware	The Charging Station SHALL send a <code>FirmwareStatusNotificationRequest</code> with status set to <code>Installed</code> or preferably to <code>InstallRebooting</code> and report another <code>FirmwareStatusNotificationRequest</code> with status <code>Installed</code> after the new firmware has been activated.	It is optional to report the <code>FirmwareStatusNotificationRequest</code> with status <code>InstallRebooting</code> , however if it is deemed necessary to report to the CSMS that the Charging Station succeeded in installing the new firmware, but needs to reboot before being able to activate the new firmware, it is recommended to use status <code>InstallRebooting</code> for this.

L03 - Publish Firmware file on Local Controller

Table 196. L03 - Publish Firmware file on Local Controller

No.	Type	Description
1	Name	Publish Firmware file on Local Controller.
2	ID	L03
	Functional block	L. FirmwareManagement
3	Objective(s)	To allow Charging Stations to download a firmware update directly from the Local Controller.

No.	Type	Description
4	Description	The Local Controller downloads and publishes a firmware update at the specified URL. This allows the CSMS to send UpdateFirmwareRequests with the URI pointing to the Local Controller, to any Charging Station connected to the Local Controller. This allows the site to save bandwidth and data on the WAN interface.
	Actors	Local Controller, CSMS
	Scenario description	<ol style="list-style-type: none"> 1. The CSMS sends a PublishFirmwareRequest to instruct the Local Controller to download and publish the firmware, including an MD5 checksum of the firmware file. 2. Upon receipt of PublishFirmwareRequest, the Local Controller responds with PublishFirmwareResponse. 3. The Local Controller starts downloading the firmware. 4. The Local Controller verifies the MD5 checksum. 5. The Local Controller publishes the firmware file at the URI(s) stated in PublishFirmwareStatusNotificationRequest. 6. The CSMS instructs Charging Stations to update their firmware, as described in Use Case L01 - Secure Firmware Update
5	Prerequisite(s)	n/a
6	Postcondition(s)	<p>Successful postcondition: The firmware is successfully published by the Local Controller.</p> <p>Failure postcondition: The Local Controller could not download the firmware file, and has sent the <i>DownloadFailed</i> status. The Local Controller could not verify the MD5 checksum, and has sent the <i>InvalidChecksum</i> status. The Local Controller could not publish the firmware file, and has sent the <i>PublishFailed</i> status.</p>

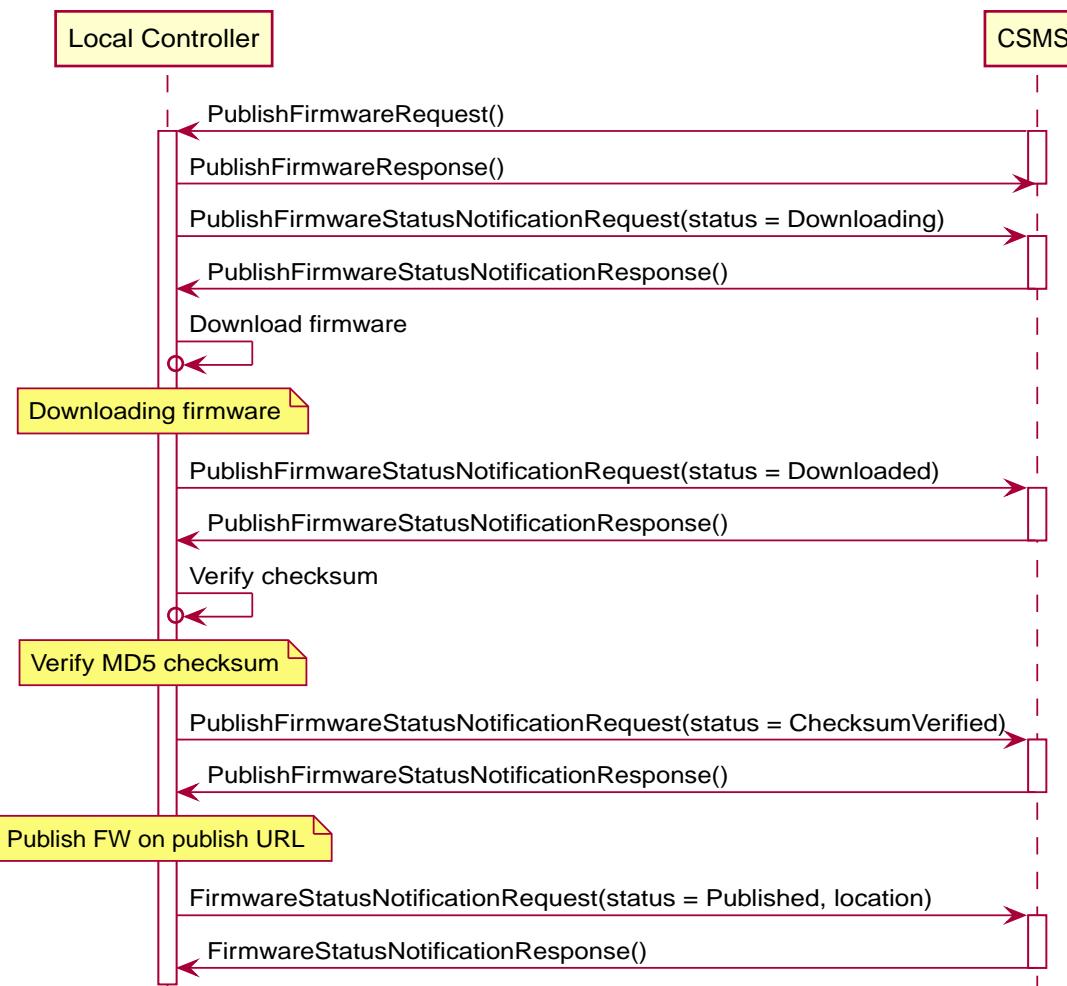


Figure 121. Sequence Diagram: showing publishing of firmware (happy flow)

7	Error handling	n/a
---	----------------	-----

8	Remark(s)	For information about MD5 checksum see RFC-1321 [RFC1321].
---	------------------	--

L03 - Publish Firmware file on Local Controller - Requirements

Table 197. L03 - Requirements

ID	Precondition	Requirement definition
L03.FR.01		Whenever the Local Controller enters a new status in the publishing process, it SHALL send a PublishFirmwareStatusNotificationRequest message to the CSMS.
L03.FR.02		The MD5 checksum SHALL be calculated over the entire firmware file.
L03.FR.03		The Local Controller SHALL publish the firmware file using all its supported protocols (e.g. HTTP, HTTPS, and FTP)
L03.FR.04		The Local Controller SHALL set URI's for all supported protocols (e.g. HTTP, HTTPS, and FTP) in the <i>location</i> field of the PublishFirmwareStatusNotificationRequest message with status <i>Published</i> .
L03.FR.05	Upon receipt of a PublishFirmwareRequest message.	The Local Controller SHALL respond with a PublishFirmwareResponse message, indicating whether it has accepted the request.
L03.FR.06	If the Local Controller cannot download the firmware file.	The Local Controller SHALL send a PublishFirmwareStatusNotificationRequest with status <i>DownloadFailed</i> .
L03.FR.07	If the Local Controller cannot verify the MD5 checksum.	The Local Controller SHALL send a PublishFirmwareStatusNotificationRequest with status <i>InvalidChecksum</i> .
L03.FR.08	If the Local Controller cannot publish the firmware file.	The Local Controller SHALL send a PublishFirmwareStatusNotificationRequest with status <i>PublishFailed</i> .
L03.FR.09	After successfully publishing the firmware file.	The Local Controller SHALL send a PublishFirmwareStatusNotificationRequest with status <i>Published</i> .
L03.FR.10	Charging Station receives a TriggerMessageRequest for PublishFirmwareStatusNotification AND last sent PublishFirmwareStatusNotificationRequest had status = Published	Charging Station SHALL return a PublishFirmwareStatusNotificationRequest with status = <i>Idle</i> .
L03.FR.11	Charging Station receives a TriggerMessageRequest for PublishFirmwareStatusNotification AND last sent PublishFirmwareStatusNotificationRequest had NOT status Published	Charging Station SHALL return a PublishFirmwareStatusNotificationRequest with the last sent status.

L04 - Unpublish Firmware file on Local Controller

Table 198. L04 - Unpublish Firmware file on Local Controller

No.	Type	Description
1	Name	Unpublish Firmware file on Local Controller.
2	ID	L04
	<i>Functional block</i>	L. FirmwareManagement
3	Objective(s)	Stop the Local Controller from publishing a firmware update to Charging Stations.
4	Description	Stop serving a firmware update to connected Charging Stations.
	Actors	Local Controller, CSMS
	Scenario description	<ol style="list-style-type: none"> 1. The CSMS sends an UnpublishFirmwareRequest to instruct the local controller to unpublish the firmware. 2. The Local Controller unpublishes the firmware. 3. The local Controller responds with an UnpublishFirmwareResponse.

No.	Type	Description
5	Prerequisite(s)	A firmware successfully published by the Local Controller.
6	Postcondition(s)	Successful postcondition: Firmware file no longer published. Failure postcondition: n/a



Figure 122. Sequence Diagram: Unpublishing a firmware file

7	Error handling	n/a
8	Remark(s)	The CSMS uses a MD5 checksum over the entire firmware file as a unique identifier to indicate which firmware file needs to be unpublished.

L04 - Unpublish Firmware file on Local Controller - Requirements

Table 199. L04 - Requirements

ID	Precondition	Requirement definition
L04.FR.01	If the Local Controller receives an UnpublishFirmwareRequest message AND There is no ongoing download.	The firmware file SHALL be unpublished.
L04.FR.02	After successfully unpublishing the firmware file.	The local controller SHALL send an UnpublishFirmwareResponse message with status <i>Unpublished</i> .
L04.FR.03	If the Local Controller receives an UnpublishFirmwareRequest message AND There is no published file.	The Local Controller SHALL send an UnpublishFirmwareResponse message with status <i>NoFirmware</i> .
L04.FR.04	If the Local Controller receives an UnpublishFirmwareRequest message AND If a Charging Station is downloading the firmware file.	The Local Controller SHALL respond with the <i>Downloading</i> status AND not unpublish the firmware file.

M. ISO 15118 CertificateManagement

1. Introduction

The ISO/IEC JWG 15118 for the Vehicle to Grid Communication Interface (V2G CI) was founded in 2009 with means to the need of a complementary international standard to IEC 61851-1 [[IEC61851-1](#)] providing bi-directional digital communication based on Internet protocols. The major purpose of 15118 is to establish a more advanced and autonomously working charge control mechanism between EVs and charging infrastructures. The standard is currently under development and will ultimately provide means for various authentication schemes (e.g. plug charge vs. external identification means, like RFID cards), automatic handling of charging services as well as (proprietary) value added services, charge scheduling and advance planning, etc.

The 15118 standard is of interest to the Open Charge Alliance, as it provides the exchange of charging schedules and enables to control the amount of power that an EV may draw from a Charging Station, in which some form of vehicle to grid communication is necessary. Especially the second part, which specifies the messages to be exchanged between the communication partners (Application Layer), the associated data and data types (Presentation Layer) via TCP/IP based Transport and Network Layer, is important to acknowledge in this specification. The authorization for charging is provided either by External Identification Means (EIM), such as an RFID card, or by the Plug and Charge (PnC) mechanism using a contract certificate stored in the EV, handled by the certificate handling process in use case elements "C", eliminating the need of other authorization means.

This 15118 OCPP Functional Block has been designed to meet a number of alignment objectives:

- To allow the communication between an EV (BEV or a PHEV) and an EVSE.
- To allow the support of certificate-based authentication and authorization at the Charging Station, i.e. plug and charge.

For illustration purposes: the figure below shows a complete sequence with authorization and scheduling.

NOTE

To the below figure: this sequence only applies for AC charging, although the certificate handling (which is the focus in this section) does not differ in AC or DC.

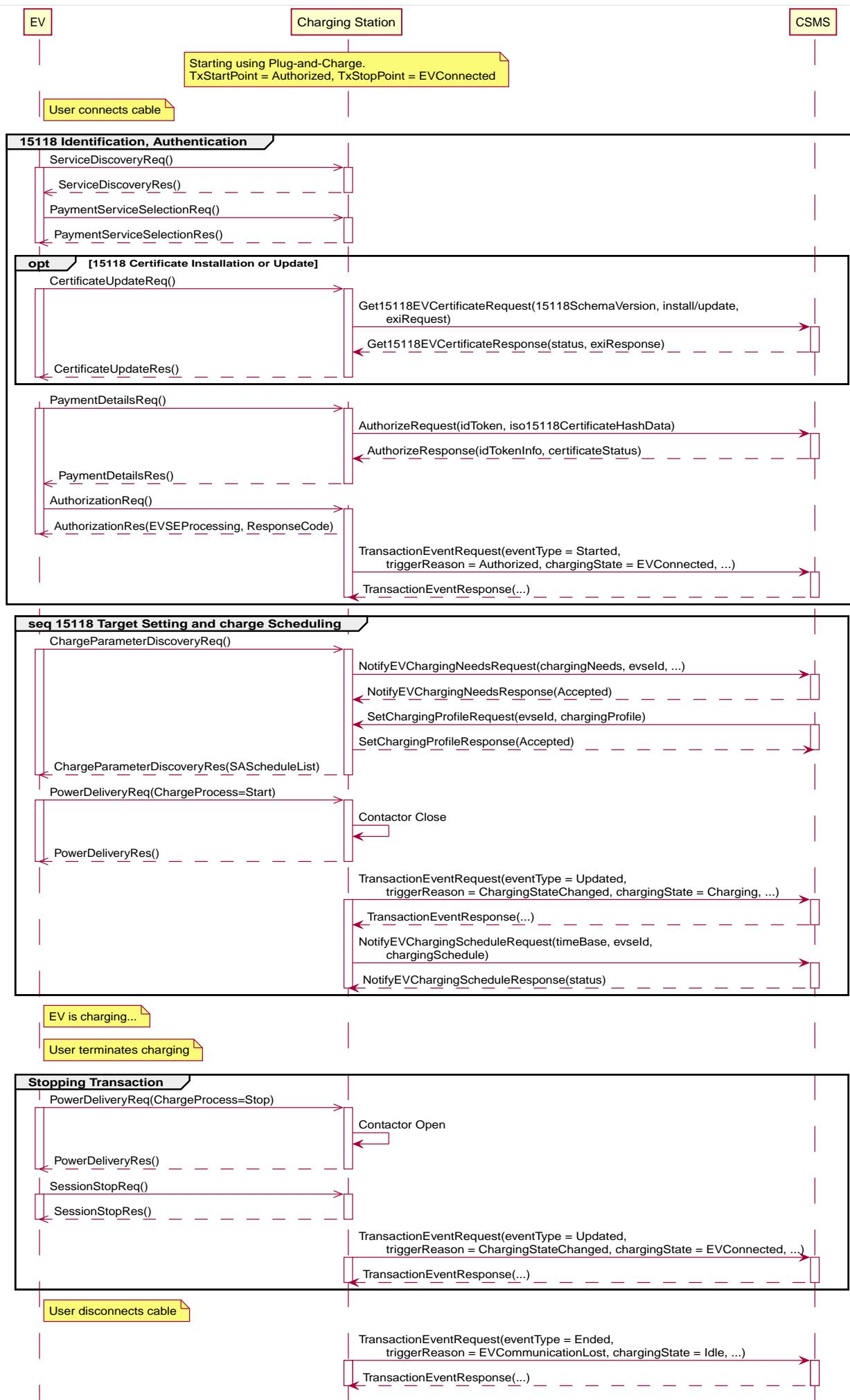


Figure 123. Sequence with Authorization and Scheduling

NOTE The time-out on the ChargeParameterDiscoveryReq is 2 seconds, but this can be prolonged up to 60 seconds to wait for charging profile to be provided by the CSMS. See ISO 15118-2 [[ISO15118-2](#)].

NOTE Please note that it is highly RECOMMENDED to use one of the TLS based security profiles from functional block A, not doing this might "break" the ISO 15118 security.

In order to control the amount of power that an EV may draw from a Charging Station, some form of vehicle to grid communication is necessary. OCPP has been designed to support the [ISO 15118](#) standard for communication between the EV and Charging Station (EVSE). However, it is anticipated that for the coming years, the majority of EVs will only support the control pilot PWM signal [IEC61851](#), so care has been taken to support smart charging with this as well.

NOTE A mapping of the ISO 15118 and OCPP terminology is provided in [ISO 15118 and OCPP terminology mapping](#) and abbreviations used in ISO 15118 are listed in [ISO 15118 Abbreviations](#).

2. ISO 15118 Certificates

2.1. ISO 15118 Certificate structure

The ISO 15118 standard provides a Plug & Charge mechanism. This is an identification and authorization mode where the customer just has to plug his electric vehicle into the EVSE and all aspects of authentication, authorization, load control and billing are automatically taken care of without the need for further user interaction. This is facilitated by the application of digital signatures and exchange of X.509 certificates bound to a Public Key Infrastructures (PKI) model.

The PKI structure defined by ISO 15118 is shown in the figure below. In general, four PKIs need to be in place.

- PKI for the Charging Station Operator (CSO)
- PKI for the Certificate Provisioning Service (CPS)
- PKI for the Mobility Operator (MO)
- PKI for the car manufacturer (OEM)

The trust anchor (root CA) for the CSO and CPS is the so-called V2G Root CA. On the other hand, it is up to the respective OEM and MO to operate a Root CA of their own or derive their certificates from a V2G Root CA (indicated by the dotted lines between V2G Root and MO Sub-CA 1 and OEM Sub-CA 1, respectively).

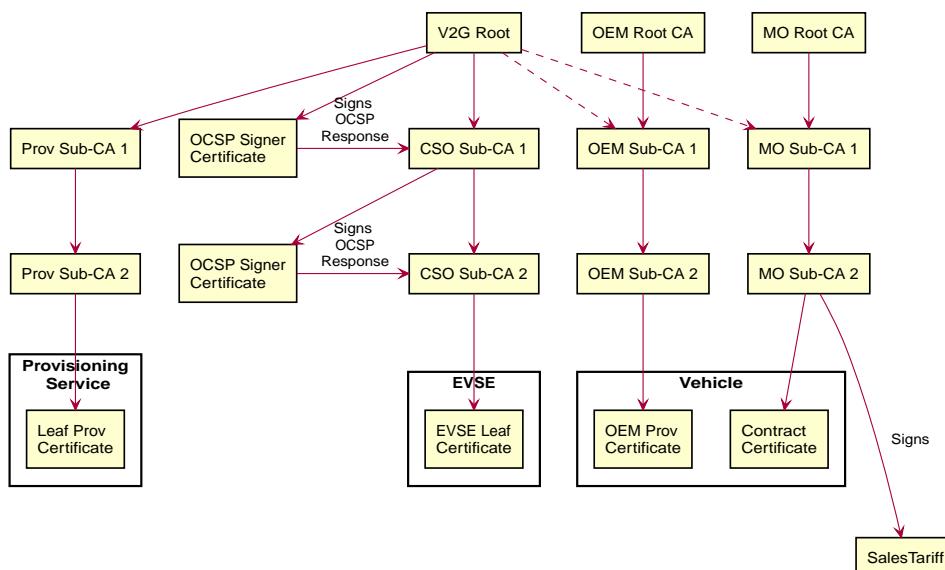


Figure 124. PKIs applied for Plug & Charge identification mode

If only one Sub-CA layer is used, i.e. a Sub-CA signed by a Root CA directly signs leaf certificates, the profile of Sub-CA 2 shall apply for that Sub-CA (Source: [ISO15118-2](#))

OCPP needs to make sure that the necessary information can be exchanged between the EV, the Charging Station and a backend IT infrastructure to facilitate the contract provisioning. Contract provisioning is a process defined within ISO 15118 that describes how an EV can retrieve a valid contract certificate during a communication session in order to authenticate and authorize itself for the charging process.

Given the PKI structure in the figure above, OCPP must provide messages which are able to transmit the following certificates:

- **CPS certificate chain**
Comprised of Prov Sub-CA 1, Prov Sub-CA 2 and leaf provisioning certificate. Sent with the CertificateInstallationRes and CertificateUpdateRes message.
- **MO certificate chain**
Comprised of MO Sub-CA 1, MO Sub-CA 2 and contract certificate. Sent with the messages CertificateInstallationRes, CertificateUpdateReq, and CertificateUpdateRes.
- **OEM provisioning certificate**
Sent with the CertificateInstallationReq message.

Furthermore, some ISO 15118 messages require digital XML-based signatures. Those signatures need to be validated by the receiving party by using the corresponding certificate chain and verifying the chain of signatures all the way up to the respective

trust anchor (V2G root, MO root or OEM root). Table 13 on page 45 of [ISO15118-2](#) provides an overview of applied XML-based signatures in ISO 15118. As you can see in there, the Charging Station (EVSE is part of a Charging Station) needs to verify the signature of the following messages.

- **AuthorizationReq**
Certificate chain needed to verify signature is provided with PaymentDetailsReq.
- **MeteringReceiptReq**
Certificate chain needed to verify signature is provided with PaymentDetailsReq.
- **CertificateUpdateReq**
Certificate chain needed to verify signature is provided with this message.

The signature verification as well as the check of the validity of each certificate provided by the EV can be done offline. These three messages are signed with the private key belonging to the public key of the contract certificate that is installed in the EV. The CSO needs to make sure that the corresponding MO root CA certificate (MO trust anchor) is installed on the Charging Station to enable signature verification offline (the chain of contract certificates and sub-CA certificates is already fulfilled by the EV in the PaymentDetailsReq message so only the MO root CA is required).

The PaymentDetailsReq message is sent before the AuthorizationReq and MeteringReceiptReq message. Therefore, the Charging Station must temporarily save the certificate chain provided with the PaymentDetailsReq message as long as the current transaction is active in order to be able to verify the signature created by the EV. After the transaction has been terminated, the temporarily saved certificate chain must be deleted on the Charging Station side.

Please note that the Charging Station only needs to check the contract certificate upon the receipt of the PaymentDetailsReq message *from* the EV which delivers the ContractSignatureCertChain, containing the contract certificate and possible sub-CA certificates, excluding the root CA certificate. However, it does not need to check the contract certificate upon installation or update of the contract certificate, upon delivery to the EV.

On the contrary, the signature provided with the **CertificateInstallationReq** needs to be verified by a so-called secondary actor, a market stakeholder communicating with the CSO backend. This means that OCPP needs to provide means for transmitting the complete CertificateInstallationReq message.

The CertificateUpdateRes and CertificateInstallationRes need to be sent from the CSO backend to the charging station as Base64 encoded binary data. The Charging Station removes the Base64 encoding and sends it to the EV as a binary EXI message.

Finally, the Charging Station certificate (labelled as EVSE Leaf Certificate in figure 1) together with its private key is used to establish a secure connection between EV and EVSE via TLS. According to ISO 15118, this certificate should be valid for only 2 to 3 months. To install or update the Charging Station certificate, please refer to [Certificate installation Charging Station](#).

While the Charging Station can verify the signature and validity period of each certificate in the MO contract certificate chain offline, there are two things which the Charging Station cannot verify offline:

1. The authorization status of the EMAID

The EMAID is a unique identifier issued by the MO together with the contract certificate. Therefore, only the MO can provide information on whether the user is authorized for charging based on this EMAID or not. The Charging Station needs to forward the EMAID to the CSO after having checked that the signature of each certificate in the contract certificate chain is valid. This order of steps is necessary because the contract certificate protects the EMAID against manipulation by means of the digital signature of its issuer. The Charging Station could also work with a white list of EMAIDs cached locally. However, white lists need to be frequently updated to ensure that the authorization information used is not outdated.

2. The revocation status of each certificate

Reasons for revoking a certificate are e.g. that the private key belonging to the public key of a certificate has been corrupted or that the algorithm used to create a signature is not considered to be secure anymore. Revocation status is checked using an OCSP responder whose address is given as an attribute value of an X.509 certificate.

2.2. Using ISO 15118 Certificates in OCPP

From an OCPP perspective, based on the above paragraph, the Charging Station needs to have one or more of each of the following certificate types:

Type	Description
V2GChargingStation Certificate	Certificate of the Charging Station. In 15118 this is called the SECC Certificate (or EVSE Leaf Certificate). This certificate is used during the set-up of the TLS connection between the Charging Station and the EV.
V2GRootCertificate	Certificate of the ISO15118 V2G Root. The V2G Charging Station Certificate MUST BE derived from this root.
MORootCertificate	Certificate from an eMobility Service provider. To support PnC charging with contracts from service providers that not derived their certificates from the V2G root.

NOTE

The V2G Charging Station Certificate might be the same as the certificate used for securing the connection between the Charging Station and the CSMS. For this to work, this certificate MUST BE to be derived from a V2G Root.

A Contract Certificate can be derived from a V2G root, or an eMobility root. This means the Charging Station needs to be in possession of the corresponding root certificate to be able to authenticate the driver by means of the Contract Certificate and the associated certificate chain.

NOTE

When a Charging Station is online this does not have to be the case, because it can send an [AuthorizeRequest](#) message with the Contract Certificate to be validated by the CSMS.

The V2G Charging Station Certificate needs to be derived from a V2G root. If this root is not known by the EV, no connection via 15118 is possible, so charging controlled by 15118 is NOT possible. In the event a Charging Station needs to support more than one V2G root, multiple V2G Charging Station Certificates are needed.

2.3. 15118 communication set-up

At the beginning of a 15118 communication session the EV will initiate a TLS Connection. In this request, the car presents its known V2G root certificates.

During the TLS handshake, the EVCC can request the OCSP status of the Charging Station and intermediate certificates using OCSP stapling as defined in [IETF RFC 6961](#). The Charging Station can retrieve this information by sending a [GetCertificateStatusRequest](#) to the CSMS, see use case [M06 - Get Charging Station Certificate status](#).

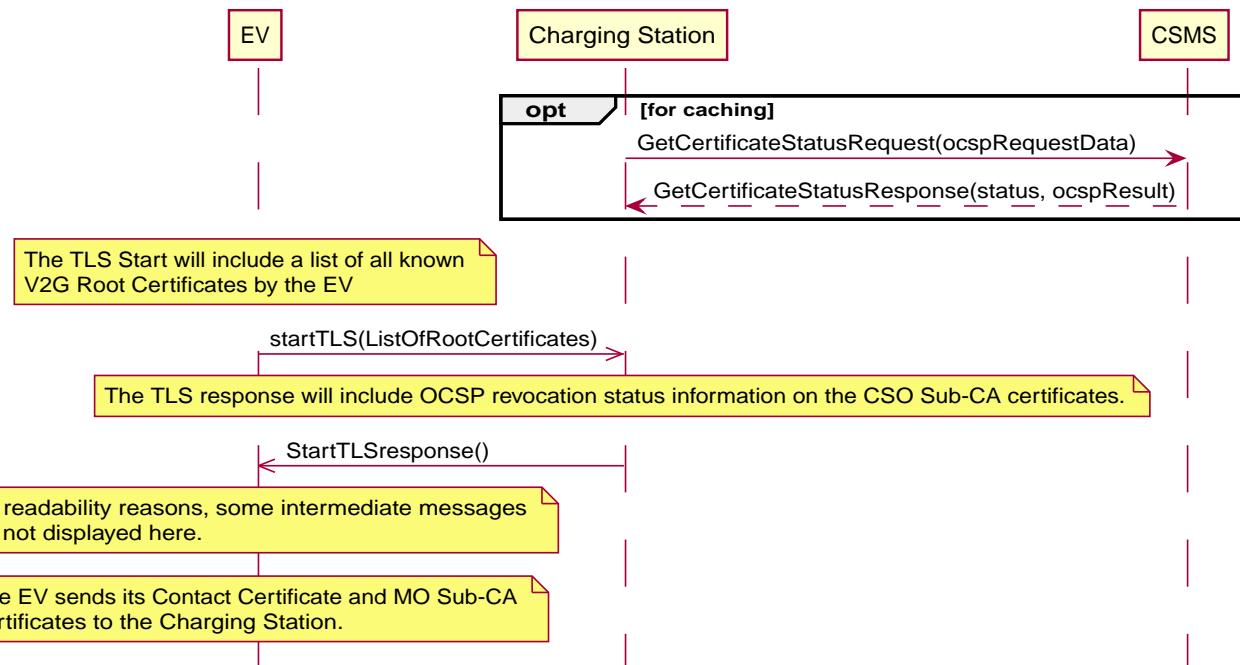


Figure 125. Communication set-up

2.4. Certificate - Use Case mapping

The following table contains the use cases that can be used to manage the certificates needed for ISO 15118 charging from OCPP:

Table 200. Certificates relevant for 15118

Certificate	Used for	Use Case	Remark
ChargingStationCertificate	Charging Station - CSMS connection	A02 and A03	Used for OCPP security in general. Certificate chain must also be available and can be retrieved by the Charging Station when installing the certificate.
CPS Certificate Chain	Plug & Charge authentication	M03, M04 and M05	
EVContractCertificate	Plug & Charge authentication	M01 and M02	Shorter life time certificate (for plug & charge)
MORootCertificate	Plug & Charge authentication	M03, M04 and M05	

Certificate	Used for	Use Case	Remark
MO Certificate Chain	Plug & Charge authentication	N.a.	It is only necessary to install MO root certificate for Plug & Charge authentication, other intermediate certificates are offered by the EV
OEMProvisioningCertificate	Installing Certificates in the EV	M01 and M02	Long life time installed in EV by OEM
V2GChargingStationCertificate	EV - Charging Station TLS connection	A02 and A03	Certificate chain must also be available and can be retrieved by the Charging Station when installing the certificate.
V2GRootCertificate	EV - Charging Station TLS connection	M03, M04 and M05	It is only necessary to install a V2G root certificate for Plug & Charge authentication.
V2GIntermediateCertificate	Plug & Charge authentication	A02, A03, M03 and M04	Intermediate certificates between the <i>V2GChargingStationCertificate</i> and <i>V2GRootCertificate</i> . May be used during TLS setup between EV and Charging Station.

3. Use cases from ISO 15118 relevant for OCPP

See [ISO15118-1](#) page 17 for a list of all elementary use cases. The **bold** indicated use case component are identified as of influence of the OCPP communication following [ISO15118-1](#).

Table 201. 15118 use cases relevant for OCPP (Source original table: [ISO15118-1](#))

No.	Use case element name / grouping
A1	Begin of charging process with forced High Level Communication
A2	Begin of charging process with concurrent IEC61851-1 and High Level Communication
B1	EV/Charging Station communication setup
C1	Certificate update
C2	Certificate installation
D1	Authorization using Contract Certificates performed at the EVSE
D2	Authorization using Contract Certificates performed with help of SA
D3	Authorization at EVSE using external credentials performed at the EVSE
D4	Authorization at EVSE using external credentials performed with help of SA
E1	AC charging with load leveling based on High Level Communication
E2	Optimized charging with scheduling to Secondary Actor
E3	Optimized charging with scheduling at EV
E4	DC charging with load leveling based on High Level Communication
E5	Resume to Authorized Charge Schedule
F0	Charging loop
F1	Charging loop with metering information exchange
F2	Charging loop with interrupt from the Charging Station
F3	Charging loop with interrupt from the EV or user
F4	Reactive power compensation
F5	Vehicle to grid support
G1	Value added services
G2	Charging details
H1	End of charging process

NOTE

Not all 15118 related OCPP use cases are described in *this* functional block. This functional block describes installing and updating certificates in the EV and CA certificate handling (also for non 15118 related purposes). Please refer to [ISO 15118 Authorization](#) for the authorization related use cases. The Smart Charging related use cases are described in the chapter [Smart Charging](#).

4. Use cases & Requirements

M01 - Certificate installation EV

Table 202. M01 - Certificate installation

No.	Type	Description
1	Name	Certificate Installation
2	ID	M01
	Functional block	M. ISO 15118 Certificate Management
	Reference	ISO15118-1 C2
3	Objectives	To install a new certificate from the CSMS in the EV.
4	Description	The EV initiates installing a new certificate. The Charging Station forwards the request for a new certificate to the CSMS. See also ISO15118-1 , use case Description C2, page 22.
	Actors	EV, Charging Station, CSMS
	Scenario description	15118: See ISO15118-1 , use case Description C2, Scenario Description, first 3 bullets, page 22. OCPP: - The Charging Station sends <code>Get15118EVCertificateRequest</code> message with <code>action = Install</code> to the CSMS. - The CSMS responds with <code>Get15118EVCertificateResponse</code> to the Charging Station.
	Alternative scenario's	n/a
5	Prerequisites	- Communication between EV and EVSE SHALL be established successfully. - Online connection between Charging Station and CSMS SHALL be possible. - CSMS should be able to communicate with a third party that can process the <code>CertificateInstallationRequest</code> , for example a contract certificate pool.
6	Postcondition(s)	See ISO15118-1 , use case End conditions C2, page 23.



Figure 126. Certificate Installation

7	Error handling	In case the CSMS is not able to respond within the specified time, the Charging Station SHALL indicate failure to the EV.
8	Remark(s)	The message timeout in ISO15118-2 for <code>CertificateInstallationReq</code> is 5 seconds. There may be alternative communication paths for doing a certificate installation. However, these are outside the scope of this standard.

Source: [ISO15118-1](#)

M01 - Certificate installation - Requirements

Table 203. M01 - Requirements

ID	Precondition	Requirement definition	Note
M01.FR.01	Upon receiving a 15118 CertificateInstallationReq	The Charging Station SHALL forward the request to the CSMS using the Get15118EVCertificateRequest message with action = Install .	The CSMS is responsible for forwarding it to the secondary actor which will process the CertificateUpdateRequest. This could be a contract certificate pool as outlined in application guide VDE-AR-2802-100-1.

M02 - Certificate Update EV

Table 204. M02 - Certificate Update

No.	Type	Description
1	Name	Certificate Update
2	ID	M02
	Functional block	M. ISO 15118 Certificate Management
	Reference	ISO15118-1 C1
3	Objectives	See ISO15118-1 , use case Objective C1, page 20.
4	Description	See ISO15118-1 , use case Description C1, page 21 up to and including the third "NOTE".
	Actors	EV, Charging Station
	Scenario description	<p>15118: See ISO15118-1, use case Objective C1, Scenario Description, first 3 bullets, page 21.</p> <p>OCPP:</p> <ul style="list-style-type: none"> - The Charging Station sends a Get15118EVCertificateRequest message with action = Update to the CSMS. - The CSMS responds with Get15118EVCertificateResponse to the Charging Station. <p>15118: See ISO15118-1, use case Description C1, Scenario Description, last 2 bullets, page 21.</p>
5	Prerequisites	<ul style="list-style-type: none"> - Communication between EV and EVSE SHALL be established successfully. - Online connection between Charging Station and CSMS SHALL be possible. - CSMS should be able to communicate with a third party that can process the CertificateInstallationRequest, for example a contract certificate pool.
6	Postcondition(s)	See ISO15118-1 , use case Objective C1 and C2, page 20/22.

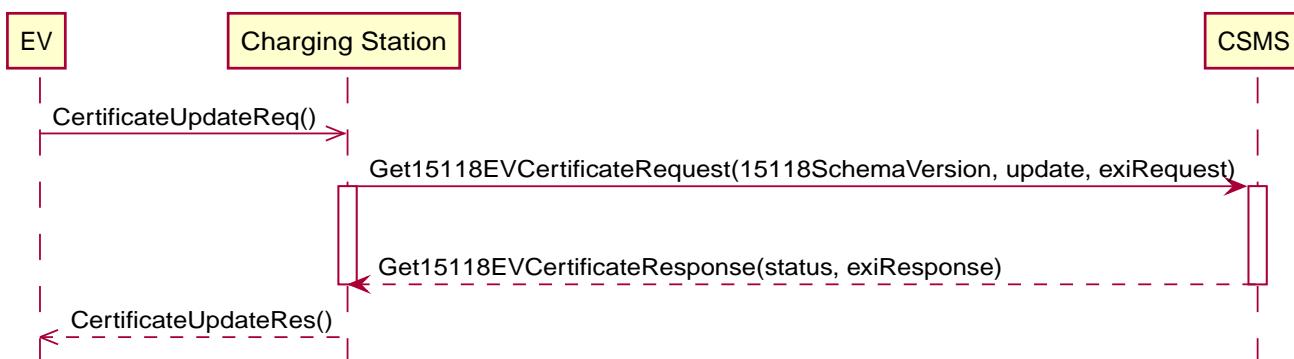


Figure 127. Certificate Update

7	Error handling	In case the CSMS is not able to respond within the specified time, the Charging Station SHALL indicate failure to the EV.
8	Remark(s)	See ISO15118-1 , use case Requirements C1, trigger , page 21. The message timeout in ISO15118-2 for CertificateUpdateReq is 5 seconds.

Source: ISO15118-1

M02 - Certificate Update - Requirements

Table 205. M02 - Requirements

ID	Precondition	Requirement definition	Note
M02.FR.01		Upon receiving a CertificateUpdateReq the Charging Station SHALL forward the request to the CSMS using the Get15118EVCertificateRequest message with action = update .	The CSMS is responsible for forwarding it to the secondary actor which will process the CertificateUpdateRequest. This could be a contract certificate pool as outlined in application guide VDE-AR-E 2802-100-1.

M03 - Retrieve list of available certificates from a Charging Station

Table 206. M03 - Retrieve list of available certificates from a Charging Station

No.	Type	Description
1	Name	Retrieve list of available certificates from a Charging Station
2	ID	M03
	<i>Functional block</i>	M. ISO 15118 Certificate Management
3	Objective(s)	To enable the CSMS to retrieve a list of available certificates from a Charging Station.
4	Description	To facilitate the management of the Charging Station's installed certificates, a method of retrieving the installed certificates is provided. The CSMS requests the Charging Station to send a list of installed certificates
	Actors	Charging Station, CSMS
	Scenario description	<p>1. The CSMS requests the Charging Station to send a list of installed certificates by sending a GetInstalledCertificateIdsRequest</p> <p>2. The Charging Station responds with a GetInstalledCertificateIdsResponse</p>
5	Prerequisite(s)	n/a
6	Postcondition(s)	The CSMS received a list of installed certificates

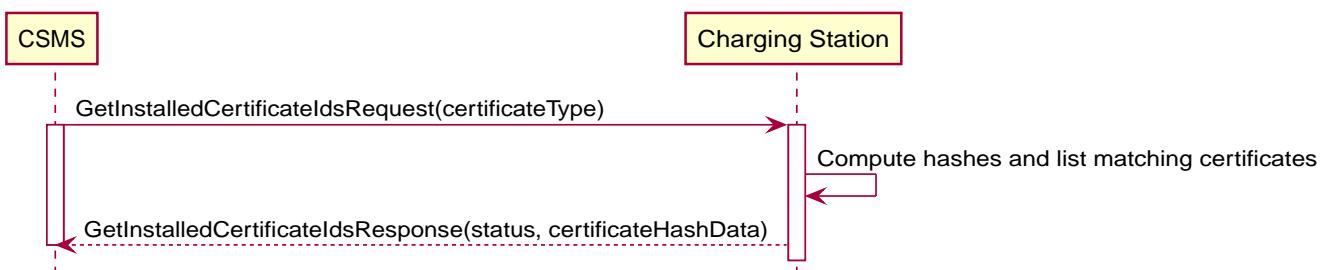


Figure 128. Retrieve list of available certificates from a Charging Station

7	Error handling	n/a
8	Remark(s)	For installing the (V2G) Charging Station Certificate, see use cases A02 - Update Charging Station Certificate by request of CSMS and A03 - Update Charging Station Certificate initiated by the Charging Station . The V2G certificate chain SHOULD not include the V2GRootCertificate. This SHOULD be installed using Use case M05 - Install CA certificate in a Charging Station .

M03 - Retrieve list of available certificates from a Charging Station - Requirements

Table 207. M03 - Requirements

ID	Precondition	Requirement definition
M03.FR.01	After receiving a GetInstalledCertificateIdsRequest	The Charging Station SHALL respond with a GetInstalledCertificateIdsResponse .

ID	Precondition	Requirement definition
M03.FR.02	M03.FR.01 AND No certificate matching certificateType was found	The Charging Station SHALL indicate this by setting status in the GetInstalledCertificateIdsResponse to <i>NotFound</i> .
M03.FR.03	M03.FR.01 AND A certificate matching certificateType was found	The Charging Station SHALL indicate this by setting status in the GetInstalledCertificateIdsResponse to <i>Accepted</i> .
M03.FR.04	M03.FR.03	The Charging Station SHALL include the hash data for each matching installed certificate in the GetInstalledCertificateIdsResponse .
M03.FR.05	When the Charging Station receives a GetInstalledCertificateIdsRequest with certificateType V2GCertificateChain	The Charging Station SHALL include the hash data for each installed certificate belonging to a V2G certificate chain. Sub CA certificates SHALL be placed as a childCertificate under the V2G Charging Station certificate.

M04 - Delete a specific certificate from a Charging Station

Table 208. M04 - Delete a specific certificate from a Charging Station

No.	Type	Description
1	Name	Delete a specific certificate from a Charging Station
2	ID	M04
	Functional block	M. ISO 15118 Certificate Management
3	Objective(s)	To enable the CSMS to request the Charging Station to delete an installed certificate.
4	Description	To facilitate the management of the Charging Station's installed certificates, a method of deleting an installed certificate is provided. The CSMS requests the Charging Station to delete a specific certificate.
	Actors	Charging Station, CSMS
	Scenario description	<ol style="list-style-type: none"> 1. The CSMS requests the Charging Station to delete an installed certificate by sending a DeleteCertificateRequest. 2. The Charging Station responds with a DeleteCertificateResponse.
5	Prerequisite(s)	n/a
6	Postcondition(s)	The requested certificate was deleted from the Charging Station.



Figure 129. Delete Installed Certificate

7	Error handling	n/a
8	Remark(s)	<p>For installing the (V2G) Charging Station Certificate, see use cases A02 - Update Charging Station Certificate by request of CSMS and A03 - Update Charging Station Certificate initiated by the Charging Station. The V2G certificate chain SHOULD not include the V2GRootCertificate. This SHOULD be installed using Use case M05 - Install CA certificate in a Charging Station.</p> <p>It is possible to delete the last (every) installed CSMSRootCertificates. When all CSMSRootCertificates are deleted, the Charging Station cannot validate CSMS Certificates, so it will not be able to connect to a CSMS. Before a CSMS would ever send a DeleteCertificateRequest that would delete the last/all CSMSRootCertificates the CSMS is ADVISED to make very sure that this is what is really wanted.</p> <p>It is possible to delete the last (every) installed ManufacturerRootCertificates, when all ManufacturerRootCertificates are deleted, no "Signed Firmware" can be installed in the Charging Station.</p>

M04 - Delete a specific certificate from a Charging Station - Requirements

Table 209. M04 - Requirements

ID	Precondition	Requirement definition	Note
M04.FR.01	After receiving a DeleteCertificateRequest	The Charging Station SHALL respond with a DeleteCertificateResponse .	
M04.FR.02	M04.FR.01 AND The requested certificate was found	The Charging Station SHALL attempt to delete it, and indicate success by setting <code>status</code> to <code>Accepted</code> in the DeleteCertificateResponse .	
M04.FR.03	M04.FR.01 AND (The deletion fails OR the Charging Station rejects the request to delete the specified certificate.)	The Charging Station SHALL indicate failure by setting <code>status</code> to <code>Failed</code> in the DeleteCertificateResponse .	A Charging Station may reject the request to prevent the deletion of a certificate, if it is the last one from its certificate type.
M04.FR.04	M04.FR.01 AND The requested certificate was not found	The Charging Station SHALL indicate failure by setting 'status' to 'NotFound' in the DeleteCertificateResponse .	
M04.FR.06	M04.FR.01 AND When <code>certificateHashData</code> refers to the <i>Charging Station Certificate</i> (see use case A)	Charging Station SHALL respond with DeleteCertificateReponse with <code>status</code> = <code>Failed</code> .	Deletion of the <i>Charging Station Certificate</i> is not allowed via DeleteCertificateRequest .
M04.FR.07	When deleting a certificate	The CSMS SHALL use the <code>hashAlgorithm</code> , which was used to install the certificate.	When a new firmware is installed it is RECOMMENDED that the CSMS requests the certificate first using GetInstalledCertificateIdsRequest to be sure of the used <code>hashAlgorithm</code> .
M04.FR.08	M04.FR.02 AND Certificate to delete is a sub-CA or root certificate	Charging Station MAY also delete all child certificates.	Else these child certificates remain as unusable orphan certificates that can no longer be deleted.

M05 - Install CA certificate in a Charging Station

Table 210. M05 - Install CA certificate in a Charging Station

No.	Type	Description
1	Name	Install CA certificate in a Charging Station
2	ID	M05
	<i>Functional block</i>	M. ISO 15118 Certificate Management
3	Objective(s)	To facilitate the management of the Charging Station's installed certificates, a method to install a new CA certificate.
4	Description	The CSMS requests the Charging Station to install a new CSMS root certificate, an eMobility Operator root certificate, Manufacturer root certificate, or a V2G root certificate.
	Actors	Charging Station, CSMS
	Scenario description	<ol style="list-style-type: none"> 1. The CSMS requests the Charging Station to install a new certificate by sending an InstallCertificateRequest. 2. The Charging Station responds with an InstallCertificateResponse.
5	Prerequisite(s)	n/a
6	Postcondition(s)	The new certificate was installed in the Charging Station trust store.

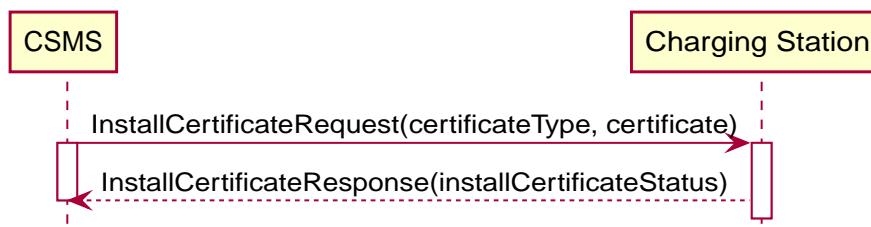


Figure 130. Install CA certificate in a Charging Station

7	Error handling	n/a
8	Remark(s)	<p>Even though the messages CertificateSignedRequest (see use cases A02 - Update Charging Station Certificate by request of CSMS and A03 - Update Charging Station Certificate initiated by the Charging Station) and InstallCertificateRequest (use case M05) are both used to send certificates, their purposes are different. CertificateSignedRequest is used to return the the Charging Stations own public certificate and V2G certificate(s) signed by a Certificate Authority. InstallCertificateRequest is used to install Root certificates.</p> <p>For installing the (V2G) Charging Station Certificate, see use cases A02 - Update Charging Station Certificate by request of CSMS and A03 - Update Charging Station Certificate initiated by the Charging Station. The V2G certificate chain SHOULD not include the V2GRootCertificate. This SHOULD be installed using this use case.</p> <p>It is allowed to have multiple certificates of the same type installed.</p>

M05 - Install CA certificate in a Charging Station - Requirements

Table 211. M05 - Requirements

ID	Precondition	Requirement definition
M05.FR.01	After receiving an InstallCertificateRequest	The Charging Station SHALL attempt to install the certificate and respond with an InstallCertificateResponse .
M05.FR.02	M05.FR.01 AND The installation was successful	The Charging Station SHALL indicate success by setting 'status' to 'Accepted' in the InstallCertificateResponse .
M05.FR.03	M05.FR.01 AND The installation failed	The Charging Station SHALL indicate failure by by setting 'status' to 'Failed' in the InstallCertificateResponse .
M05.FR.06	When a new certificate gets installed AND the CertificateEntries.maxLimit is going to be exceeded	The Charging Station SHALL respond with status <i>Rejected</i> .
M05.FR.07	M05.FR.01 AND The certificate is invalid.	The Charging Station SHALL indicate rejection by setting 'status' to 'Rejected' in the InstallCertificateResponse .
M05.FR.09	When AdditionalRootCertificateCheck is true	Only one certificate (plus a temporarily fallback certificate) of certificateType CSMSRootCertificate is allowed to be installed at a time.
M05.FR.10	When AdditionalRootCertificateCheck is true AND installing a new certificate of certificateType CSMSRootCertificate	The new CSMS Root certificate SHALL replace the old CSMS Root certificate AND the new Root Certificate MUST be signed by the old Root Certificate it is replacing
M05.FR.11	M05.FR.10 AND the new CSMS Root certificate is NOT signed by the old CSMS Root certificate	The Charging Station SHALL NOT install the new CSMS Root Certificate and respond with status <i>Rejected</i> .
M05.FR.12	M05.FR.10 AND the new CSMS Root certificate is signed by the old CSMS Root certificate	The Charging Station SHALL install the new CSMS Root Certificate AND temporarily keep the old CSMS Root certificate as a fallback certificate AND respond with status <i>Accepted</i>
M05.FR.13	M05.FR.12 AND the Charging Station successfully connected to the CSMS using the new CSMS Root certificate	The Charging Station SHALL remove the old CSMS Root (fallback) certificate.
M05.FR.14	M05.FR.12 AND The Charging Station is attempting to reconnect to the CSMS (NOT migrating to another CSMS with Use Case B10 - Migrate to new CSMS), but determines that the server certificate provided by the CSMS is invalid when using the new CSMS Root certificate to verify it	The Charging Station SHALL try to use the old CSMS Root (fallback) certificate to verify the server certificate.

ID	Precondition	Requirement definition
M05.FR.15	M05.FR.12 AND When the Charging Station is migrating to another CSMS with Use Case B10 - Migrate to new CSMS , but determines that the server certificate provided by the CSMS is invalid when using the new CSMS Root certificate to verify it	The Charging Station SHALL use the NetworkProfileConnectionAttempts mechanism as described at Use Case B10 - Migrate to new CSMS .
M05.FR.16	M05.FR.15 AND If after the number of attempts the connection fails AND If it goes back to the old NetworkConnectionProfile (See B10.FR.03)	The Charging Station SHALL use the old CSMS Root (fallback) certificate to verify the server certificate.
M05.FR.17	NOT M05.FR.10 AND After receiving an InstallCertificateRequest for a certificate that is already present in the certificate trust store of the Charging Station	The Charging Station SHALL replace the certificate and respond with InstallCertificateResponse with status = Accepted.

M06 - Get V2G Charging Station Certificate status

Table 212. M06 - Get V2G Charging Station Certificate status

No.	Type	Description
1	Name	Get V2G Charging Station Certificate status
2	ID	M06
	Functional block	M. ISO 15118 Certificate Management
3	Objective(s)	To enable a Charging Station to cache the OCSP certificate status needed for the TLS handshake between EV and Charging Station.
4	Description	When the cable gets plugged in and an ISO 15118 supported EV gets connected to the Charging Station, the EV requests the Charging Station to prove the validity of the (SubCA) certificates by an OCSPResponse. A request needs to be sent per SubCA. Because the timeout constraint in ISO 15118 is too strict to make the call to an external server, OCPP requires to cache the OCSP certificate status of the certificates beforehand. The Charging Station needs to refresh the cached OCSP data once a week..
	Actors	Charging Station, CSMS
	Scenario description	<ol style="list-style-type: none"> 1. The Charging Station requests the CSMS to provide OCSP certificate status by sending a GetCertificateStatusRequest. 2. The CSMS responds with a GetCertificateStatusResponse.
5	Prerequisite(s)	n/a
6	Postcondition(s)	<p>Successful postcondition: The Charging Station received the OCSP certificate status for the requested certificate</p> <p>Failure postcondition: The retrieval of the OCSP certificate status by the CSMS failed</p>

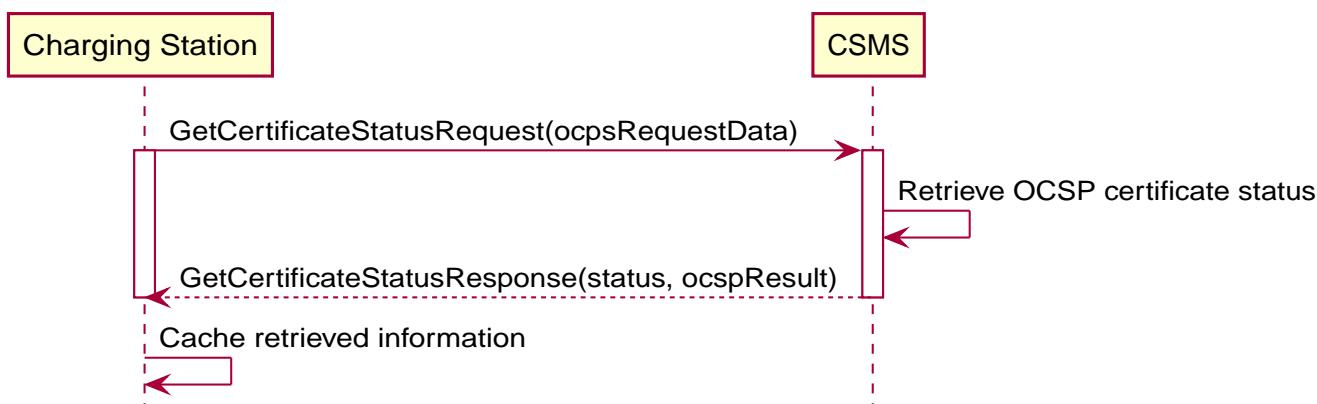


Figure 131. Get V2G Charging Station Certificate status

7	Error handling	n/a
8	Remark(s)	<p>The status indicator in the GetCertificateStatusResponse indicates whether or not the CSMS was successful in retrieving the certificate status. it does NOT indicate the validity of the certificate.</p> <p>For installing the (V2G) Charging Station Certificate, see use cases A02 - Update Charging Station Certificate by request of CSMS and A03 - Update Charging Station Certificate initiated by the Charging Station. The V2G certificate chain SHOULD not include the V2GRootCertificate. This SHOULD be installed using Use case M05 - Install CA certificate in a Charging Station.</p> <p>OCPP allows for only one certificate per GetCertificateStatusRequest. Because when multiple answers on a GetCertificateStatusRequest are to be expected, it makes handling the request and status more complex. So a GetCertificateStatusRequest needs to be sent per SubCA.</p> <p><i>responderURL</i> is required in OCPP, while it is optional in ISO 15118. Without a <i>responderURL</i> in a certificate it cannot work, so a <i>responderURL</i> is required for any certificate for which a GetCertificateStatusRequest can be expected.</p>

M06 - Get V2G Charging Station Certificate status - Requirements

Table 213. M06 - Requirements

ID	Precondition	Requirement definition
M06.FR.01	After receiving a GetCertificateStatusRequest	The CSMS SHALL respond with a GetCertificateStatusResponse .
M06.FR.02	M06.FR.01 AND The CSMS was successful in retrieving the OCSP certificate status	The CSMS SHALL indicate success by setting 'status' to 'Accepted' in the GetCertificateStatusResponse .
M06.FR.03	M06.FR.02	The CSMS SHALL include the OCSP response data in the OCSPResult field in the GetCertificateStatusResponse .
M06.FR.04	M06.FR.01 AND The CSMS was not successful in retrieving the OCSP certificate status	The CSMS SHALL indicate it was not successful by setting <i>status</i> to Failed in the GetCertificateStatusResponse .
M06.FR.06		The Charging Station SHALL request and cache the OCSP status for its V2G certificates.
M06.FR.07		After the Charging Station Certificate has been updated, The Charging Station SHALL refresh the cached OCSP data by sending a GetCertificateStatusRequest for the new certificate, and also for the intermediate certificates.
M06.FR.08		The CSMS SHALL format the response data according to OCSPResponse as defined in IETF RFC 6960 , formatted according to ASN.1 [X.680].
M06.FR.09		The OCSPResponse data SHALL be DER encoded.
M06.FR.10		The Charging Station SHALL refresh the cached OCSP data at least once a week.

N. Diagnostics

1. Introduction

This Functional Block describes the diagnostics functionality of OCPP. This functionality enables remote diagnostics of problems with a Charging Station. A Charging Station can be requested to upload a file with diagnostics information (optionally limited to a specified interval).

2. Use cases & Requirements

2.1. Logging

N01 - Retrieve Log Information

Table 214. N01 - Retrieve Log Information

No.	Type	Description
1	Name	Retrieve Log
2	ID	N01
	Functional block	N. Diagnostics
3	Objective(s)	To enable the CSMS retrieving of log information from a Charging Station.
4	Description	This use case covers the functionality of getting log information from a Charging Station. The CSMS can request a Charging Station to upload a file with log information to a given location (URL). The format of this log file is not prescribed. The Charging Station uploads a log file and gives information about the status of the upload by sending status notifications to the CSMS.
	Actors	Charging Station, CSMS
	Scenario description	<ol style="list-style-type: none"> 1. The CSMS sends a GetLogRequest to the Charging Station. 2. The Charging Station responds with a GetLogResponse. 3. The Charging Station sends a LogStatusNotificationRequest with the status <code>Uploading</code> 4. The CSMS responds with a LogStatusNotificationResponse acknowledging the status update request. 5. Uploading of the diagnostics files. 6. The Charging Station sends LogStatusNotificationRequest with the status <code>Uploaded</code>. 7. The CSMS responds with LogStatusNotificationResponse, acknowledging the status update request.
5	Prerequisite(s)	<ul style="list-style-type: none"> - Diagnostics information is available for upload. - URL to upload file to is reachable and exists.
6	Postcondition(s)	<p>Successful postcondition: Log file successfully uploaded.</p> <p>Failure postcondition: Log file not successfully uploaded and failed.</p>

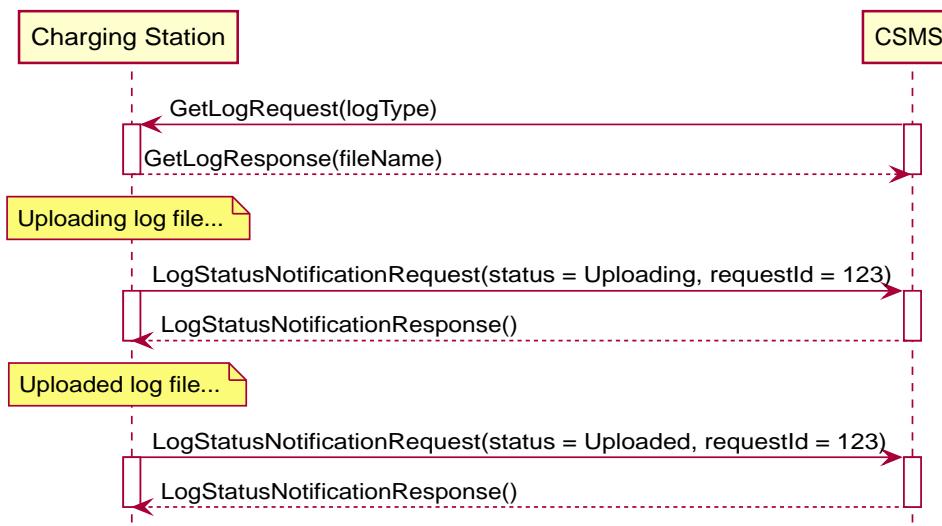


Figure 132. Sequence Diagram: Get Diagnostics

7	Error handling	When the upload fails and the transfer protocol supports "resume" the Charging Station is RECOMMENDED to try to resume before aborting the upload.
---	----------------	--

8	Remark(s) <p>As an example in this use case the requestId = 123, but this could be any value.</p> <p>When a Charging Station is requested to upload a log file, the CSMS supplies in the request an URL where the Charging Station SHALL upload the file. The URL also contains the protocol which must be used to upload the file.</p> <p>It is recommended that the log file is uploaded via FTP or FTPS. FTP(S) is better optimized for large binary data than HTTP. Also FTP(S) has the ability to resume uploads. In case an upload is interrupted, the Charging Station can resume uploading after the part it already has uploaded. The FTP URL is of format: <code>ftp://User:password@host:port/path</code> in which the parts <code>User:password@</code>, <code>:password</code> or <code>:port</code> may be excluded.</p> <p>The Charging Station has a required Configuration Variable that reports which file transfer protocols it supports: FileTransferProtocols</p> <p>The format of the log file is not prescribed.</p> <p>FTP needs to be able to use Passive FTP, to be able to transverse over as much different typologies as possible.</p>
---	---

N01 - Retrieve Log Information - Requirements

Table 215. N01 - Requirements

ID	Precondition	Requirement definition	Note
N01.FR.01	Upon receipt of a GetLogRequest AND if the requested log information is available	The Charging Station SHALL respond with a GetLogResponse stating the name of the file and status Accepted.	
N01.FR.02	N01.FR.01	The Charging Station SHALL start uploading a single log file to the specified location	
N01.FR.03	N01.FR.02 AND The GetLogRequest contained logType <i>SecurityLog</i>	The Charging Station SHALL upload its security log	
N01.FR.04	N01.FR.02 AND The GetLogRequest contained logType <i>DiagnosticsLog</i>	The Charging Station SHALL upload its diagnostics.	
N01.FR.05	Upon receipt of a GetLogRequest AND if the requested log information is NOT available	The Charging Station SHALL respond with a GetLogResponse WITH status Rejected.	
N01.FR.07		Every LogStatusNotificationRequest sent for a log upload SHALL contain the same requestId as the GetLogRequest that started this log upload.	
N01.FR.08	When uploading a log document is started	The Charging Station SHALL send a LogStatusNotificationRequest with status <i>Uploading</i> .	
N01.FR.09	When a log document is uploaded successfully	The Charging Station SHALL send a LogStatusNotificationRequest with status <i>Uploaded</i> .	
N01.FR.10	When uploading a log document failed	The Charging Station SHALL send a LogStatusNotificationRequest with status <i>UploadFailure</i> , <i>BadMessage</i> , <i>PermissionDenied</i> OR <i>NotSupportedOperation</i> .	It is RECOMMENDED to send a status that describes the reason of failure as precise as possible.
N01.FR.12	When a Charging Station is assembling or uploading the log file AND the Charging Station receives a new GetLogRequest	The Charging Station SHOULD cancel the ongoing log file upload AND respond with status <i>AcceptedCanceled</i> .	

ID	Precondition	Requirement definition	Note
N01.FR.13		The field requestId in LogStatusNotificationRequest is mandatory, unless the message was triggered by a TriggerMessageRequest AND there is no log upload ongoing.	
N01.FR.14		It is RECOMMENDED that Charging Station and CSMS support at least HTTP(s) as transport mechanism for the log file upload	HTTP transport is most likely to be supported, since it is also used for OCPP messaging.
N01.FR.15		Charging Station SHALL at least support the CSMS trust chain for secure transports	
N01.FR.16		It is RECOMMENDED that Charging Station supports the usual CAs provided by the operating system	The log file storage of CSMS may be a cloud service operated separately from the CSMS itself and not part of the CSMS trustchain.
N01.FR.17	When CSMS requires basic authorization for the upload	CSMS is RECOMMENDED to require a different basic authorization password for the upload, than the one used for OCPP connectivity.	This is to avoid leaking the OCPP password to 3rd parties if the log file storage is a different system. Basic authorization can be added to the URL as follows: http://username:password@csms.org/logs
N01.FR.18		Is is RECOMMENDED that CSMS accepts both PUT and POST requests for uploads from Charging Station.	
N01.FR.19	When Charging Station uses a HTTP(s) POST request to upload the log file	Charging Station SHALL provide at least the following attributes: Content-Type : (e.g. application/octet-stream) and Content-Disposition : with a specification of the filename.	For example: Content-Type: application/octet-stream Content-Disposition: form-data; name="uploadedfile"; filename="logfile_20210420.zip"
N01.FR.20	N01.FR.12 AND Charging Station cancels the log file upload	The Charging Station SHALL send a LogStatusNotificationRequest with status = AcceptedCanceled.	N01.FR.12 is a "SHOULD" requirement. Only send status notification when requirement is executed.

2.2. Configure Monitoring

NOTE

For managing the monitoring of a Charging Station a basic understanding of Device Model concepts is essential. These concepts are explained in "OCPP 2.0.1: Part 1 - Architecture & Topology", chapter 4.

N02 - Get Monitoring report

Table 216. N02 - Get Monitoring Report

No.	Type	Description
1	Name	Get Monitoring Report
2	ID	N02
	Functional block	N. Diagnostics
3	Objective(s)	To give the CSMS the ability to retrieve a report about configured monitoring settings per component and variable.
4	Description	This use case describes how the CSMS requests the Charging Station to send a report about configured monitoring settings per component and variable. Optionally, this list can be filtered on monitoringCriteria and componentVariables.

No.	Type	Description
	Actors	Charging Station, CSMS, CSO
	Scenario description	<p>1. The CSO triggers the CSMS to request a monitoring report from a Charging Station.</p> <p>2. The CSMS sends a GetMonitoringReportRequest to the Charging Station.</p> <p>3. The Charging Station responds with a GetMonitoringReportResponse.</p> <p>4. The Charging Station sends a NotifyMonitoringReportRequest to the CSMS.</p> <p>5. The CSMS responds with a NotifyMonitoringReportResponse.</p> <p>6. Steps #4 and #5 are repeated until all data of the monitoring report has been sent.</p>
5	Prerequisite(s)	Charging Station supports Monitoring
6	Postcondition(s)	The CSMS received a report about the configured monitoring settings.

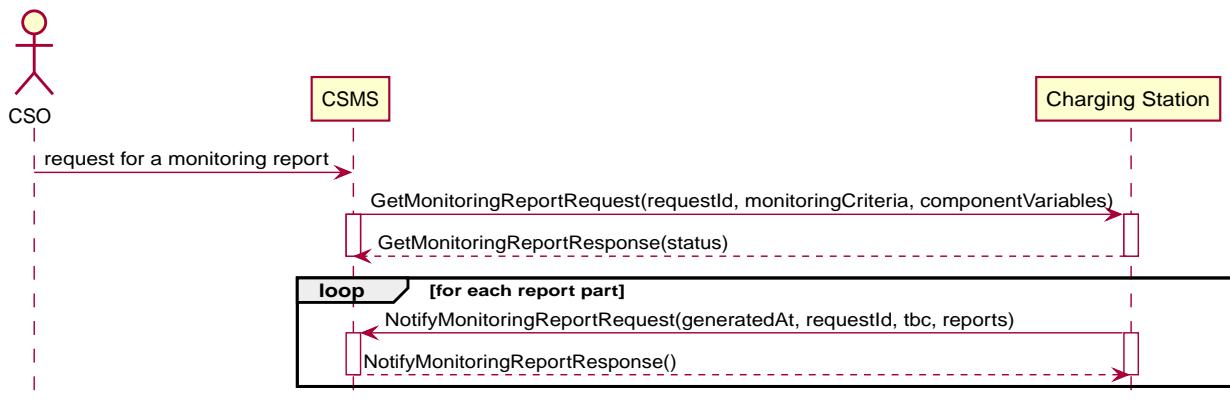


Figure 133. Sequence Diagram: Get Monitoring Report

7	Error handling	n/a
8	Remark(s)	n/a

N02 - Get Monitoring Report - Requirements

Table 217. N02 - Requirements

ID	Precondition	Requirement definition
N02.FR.01	NOT N02.FR.10 AND When the Charging Station receives a getMonitoringReportRequest for supported <i>monitoringCriteria</i> OR without <i>monitoringCriteria</i>	The Charging Station SHALL send a getMonitoringReportResponse with Accepted .
N02.FR.02	When the Charging Station receives a getMonitoringReportRequest for not supported <i>monitoringCriteria</i>	The Charging Station SHALL send a getMonitoringReportResponse with NotSupported .
N02.FR.03	N02.FR.01	The Charging Station SHALL send the requested information via one or more notifyMonitoringReportRequest messages to the CSMS.
N02.FR.04	N02.FR.01 AND The getMonitoringReportRequest contained a <i>requestId</i>	Every notifyMonitoringReportRequest sent for this getMonitoringReportRequest SHALL contain the same <i>requestId</i> .
N02.FR.05	N02.FR.01 AND <i>monitoringCriteria</i> and <i>componentVariables</i> are NOT both empty.	The set of monitors reported in one or more notifyMonitoringReportRequest messages is limited to the set defined by <i>monitoringCriteria</i> and <i>componentVariables</i> .
N02.FR.06	N02.FR.01 AND <i>monitoringCriteria</i> is NOT empty AND <i>componentVariables</i> is empty.	The set of monitors reported in one or more notifyMonitoringReportRequest messages is limited to the set defined by <i>monitoringCriteria</i> .
N02.FR.07		The maximum number of <i>componentVariables</i> in one getMonitoringReportRequest message is given by the ItemsPerMessageGetReport Configuration Variable
N02.FR.08	N02.FR.01 AND <i>monitoringCriteria</i> is absent AND <i>componentVariables</i> is NOT empty.	The set of monitors reported in one or more notifyMonitoringReportRequest messages is limited to the set defined by <i>componentVariables</i> .

ID	Precondition	Requirement definition
N02.FR.09		The sequence number contained in the seqNo field of the NotifyMonitoringReportRequest is incremental per report. So the NotifyMonitoringReportRequest message which contains the first report part, SHALL have a seqNo with value 0.
N02.FR.10	When the Charging Station receives a GetMonitoringReportRequest with a combination of criteria which results in an empty result set.	The Charging Station SHALL respond with a GetMonitoringReportResponse (status=EmptyResultSet).
N02.FR.11	N02.FR.01 AND <i>monitoringCriteria</i> is empty AND <i>componentVariables</i> is empty.	The set of all existing monitors is reported in one or more notifyMonitoringReportRequest messages.
N02.FR.12	If <i>monitoringCriteria</i> contains ThresholdMonitoring	All monitors with <i>type</i> = UpperThreshold or <i>type</i> = LowerThreshold are reported.
N02.FR.13	If <i>monitoringCriteria</i> contains DeltaMonitoring	All monitors with <i>type</i> = Delta are reported.
N02.FR.14	If <i>monitoringCriteria</i> contains PeriodicMonitoring	All monitors with <i>type</i> = Periodic or <i>type</i> = PeriodicClockAligned are reported.
N02.FR.15	When Charging Station receives a GetMonitoringReportRequest with <i>componentVariable</i> elements in which <i>component.instance</i> and/or <i>component.evse</i> are missing	The Charging Station SHALL report for every instance and/or EVSE of the component in <i>componentVariable</i> .
N02.FR.16	When Charging Station receives a GetMonitoringReportRequest with <i>componentVariable</i> elements in which <i>variable</i> is missing	The Charging Station SHALL report for every variable of the component in <i>componentVariable</i> .
N02.FR.17	When Charging Station receives a GetMonitoringReportRequest with <i>componentVariable</i> elements in which <i>variable</i> is present, but <i>instance</i> is missing	The Charging Station SHALL report for every instance of the variable of the component in <i>componentVariable</i> .

N03 - Set Monitoring Base

Table 218. N03 - Set Monitoring Base

No.	Type	Description
1	Name	Set Monitoring Base
2	ID	N03
	<i>Functional block</i>	N. Diagnostics
3	Objective(s)	To give the CSMS the ability to request the Charging Station to activate a set of preconfigured monitoring settings, as denoted by the value of MonitoringBase .
4	Description	This use case describes how the CSMS requests the Charging Station to activate a set of preconfigured monitoring settings, as denoted by the value of MonitoringBase . It is up to the manufacturer of the Charging Station to define which monitoring settings are activated by All, FactoryDefault and HardWiredOnly.
	Actors	Charging Station, CSMS, CSO
	Scenario description	<ol style="list-style-type: none"> 1. The CSO triggers the CSMS to request a Charging Station to set a monitoring base. 2. The CSMS sends a SetMonitoringBaseRequest to the Charging Station. 3. The Charging Station responds with a SetMonitoringBaseResponse.
5	Prerequisite(s)	Charging Station supports Monitoring
6	Postcondition(s)	The Charging Station activated the set of monitoring settings, as denoted by the value of MonitoringBase .

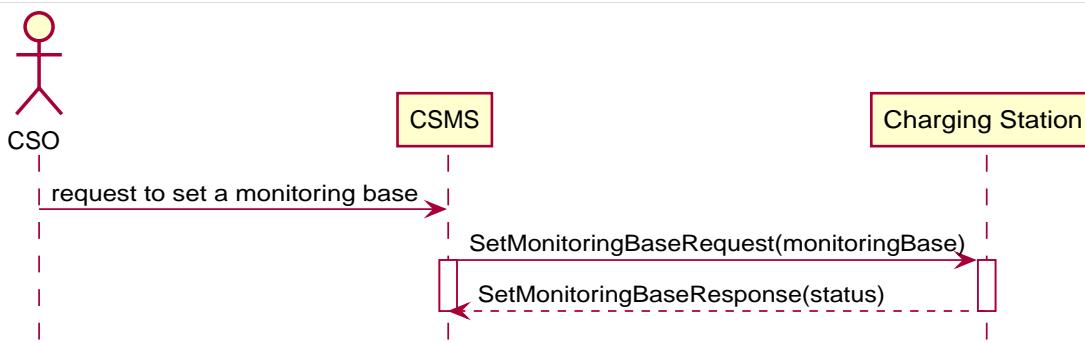


Figure 134. Sequence Diagram: Set Monitoring Base

7	Error handling	n/a
8	Remark(s)	<p>Upon receipt of a SetMonitoringBaseRequest for HardWiredOnly or FactoryDefault the Charging Station will discard of any previously configured custom monitors and will activate the monitoring settings that are related to given MonitoringBase.</p> <p>For a MonitoringBase = All the Charging Station will activate all pre-configured monitors and leave previously configured custom monitors intact. This includes the custom monitors that were created when changing an existing pre-configured monitor.</p> <p>When the set of pre-configured monitors for All and FactoryDefault is the same, then the difference between the two is, that with FactoryDefault all custom monitors are deleted before the factory default pre-configured monitors are restored.</p>

N03 - Set Monitoring Base - Requirements

Table 219. N03 - Requirements

ID	Precondition	Requirement definition
N03.FR.01	When the Charging Station accepts a <code>setMonitoringBaseRequest</code>	Then the Charging Station SHALL send a <code>setMonitoringBaseResponse</code> with Accepted.
N03.FR.02	When the Charging Station receives a <code>setMonitoringBaseRequest</code> for a not supported <code>monitoringBase</code>	Then the Charging Station SHALL send a <code>setMonitoringBaseResponse</code> with NotSupported.
N03.FR.03	N03.FR.01 AND When the Charging Station received a <code>setMonitoringBaseRequest</code> with <code>monitoringBase All</code>	Then the Charging Station SHALL activate all preconfigured monitoring whilst leaving all installed custom monitors (including changed preconfigured monitors) intact.
N03.FR.04	N03.FR.01 AND When the Charging Station received a <code>setMonitoringBaseRequest</code> with <code>monitoringBase FactoryDefault</code>	Then the Charging Station SHALL delete all custom monitors (including overruled pre-configured monitors) and activate the default monitoring settings as recommended by the manufacturer.
N03.FR.05	N03.FR.01 AND When the Charging Station received a <code>setMonitoringBaseRequest</code> with <code>monitoringBase HardWiredOnly</code>	Then the Charging Station SHALL clear all custom and disable all pre-configured monitors. Only hard-wired monitors remain active.

N04 - Set Variable Monitoring

Table 220. N04 - Set Variable Monitoring

No.	Type	Description
1	Name	Set Variable Monitoring
2	ID	N04
	Functional block	N. Diagnostics
3	Objective(s)	To give the CSMS the ability to request the Charging Station to set monitoring triggers on Variables.

No.	Type	Description
4	Description	This use case describes how the CSMS requests the Charging Station to set monitoring triggers on Variables. Multiple triggers can be set for upper or lower thresholds, delta changes or periodic reporting.
	Actors	Charging Station, CSMS, CSO
	Scenario description	<ol style="list-style-type: none"> 1. The CSO triggers the CSMS to request a Charging Station to set a variable monitoring setting. 2. The CSMS sends a SetVariableMonitoringRequest to the Charging Station. 3. The Charging Station responds with a SetVariableMonitoringResponse.
5	Prerequisite(s)	Charging Station supports Monitoring The specific Variable supports Monitoring
6	Postcondition(s)	The Charging Station activated the set of monitoring triggers on the Variables.

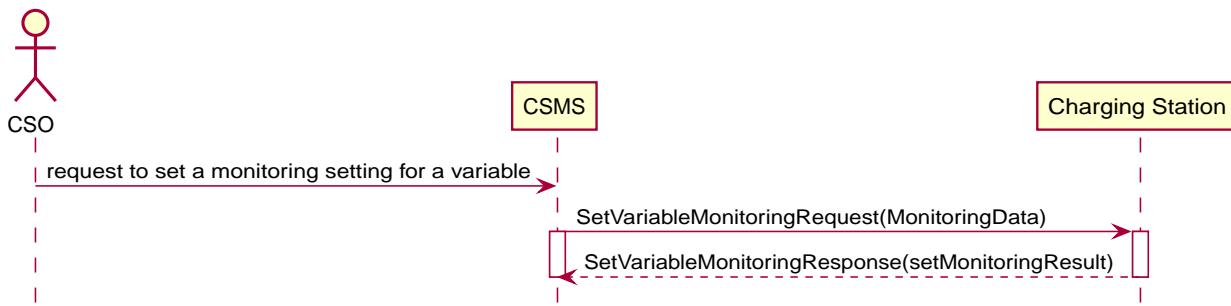


Figure 135. Sequence Diagram: Set Variable Monitoring

7	Error handling	n/a
8	Remark(s)	All variableMonitoring settings are persistent across reboot. A variableMonitoring setting is persistent after a firmware update, if the monitored variable still exists and it is still monitorable. Otherwise the variableMonitoring setting is removed.

N04 - Set Variable Monitoring - Requirements

Table 221. N04 - Requirements

ID	Precondition	Requirement definition	Note
N04.FR.01	When the Charging Station receives a <code>SetVariableMonitoringRequest</code> with an X number of <code>SetMonitoringData</code> elements	The Charging Station SHALL respond with an <code>SetVariableMonitoringResponse</code> with an equal (X) number of <code>SetMonitoringResult</code> elements, one for every <code>SetMonitoringData</code> element in the <code>SetVariableMonitoringRequest</code> .	
N04.FR.02	N04.FR.01	Every <code>SetMonitoringResult</code> element in the <code>SetVariableMonitoringResponse</code> SHALL contain the same <code>component</code> and <code>variable</code> combination as one of the <code>SetVariableMonitoringRequest</code> elements in the <code>SetVariableMonitoringRequest</code> .	
N04.FR.03	When the Charging Station receives a <code>SetVariableMonitoringRequest</code> with an unknown Component in <code>SetMonitoringData</code>	The Charging Station SHALL set the <code>attributeStatus</code> field in the corresponding <code>SetMonitoringResult</code> to: <code>UnknownComponent</code> .	
N04.FR.04	When the Charging Station receives a <code>SetVariableMonitoringRequest</code> with a Variable that is unknown for the given Component in <code>SetMonitoringData</code>	The Charging Station SHALL set the <code>attributeStatus</code> field in the corresponding <code>SetMonitoringResult</code> to: <code>UnknownVariable</code> .	
N04.FR.05	When the Charging Station receives a <code>SetVariableMonitoringRequest</code> with an MonitorType which is not supported by the specific Variable	The Charging Station SHALL set the <code>attributeStatus</code> field in the corresponding <code>SetMonitoringResult</code> to: <code>UnsupportedMonitorType</code> .	
N04.FR.06	When the Charging Station receives a <code>SetVariableMonitoringRequest</code> with monitor type <code>UpperThreshold</code> or <code>LowerThreshold</code> AND the <code>monitorValue</code> is lower or higher than the range of the given Variable	The Charging Station SHALL set the <code>attributeStatus</code> field in the corresponding <code>SetMonitoringResult</code> to: <code>Rejected</code> .	More information can be provided in the optional <code>statusInfo</code> element.

ID	Precondition	Requirement definition	Note
N04.FR.07	When the Charging Station receives a SetVariableMonitoringRequest for a monitor that conflicts with safety requirements.	The Charging Station MAY set the <i>attributeStatus</i> field in the corresponding SetMonitoringResult to: Rejected .	e.g. when the requested monitoring overrides factory set security monitoring.
N04.FR.08	When the Charging Station was able to set the given <i>monitorValue</i> in the SetMonitoringData	The Charging Station SHALL set the <i>attributeStatus</i> field in the corresponding SetMonitoringResult to: Accepted .	Please refer to use case N07 - Alert Event on how to handle the different monitor types .
N04.FR.09		The maximum size and number of items of <i>monitoringData</i> in one SetVariableMonitoringRequest message is determined by the ItemsPerMessageSetVariableMonitoring and BytesPerMessageSetVariableMonitoring Configuration Variables.	
N04.FR.10	When the Charging Station receives a SetVariableMonitoringRequest for a <i>component/variable</i> combination for which a monitor with the same type and severity already exists with a different <i>id</i> .	The Charging Station SHALL set the <i>attributeStatus</i> field in the corresponding SetMonitoringResult to: Duplicate .	There cannot be two monitors of the same type with the same severity on the same variable. E.g. when a component/variable has a monitor with an <i>UpperThreshold</i> at value "67" and severity "4-Error", then there cannot be another <i>UpperThreshold</i> at value "78" with same severity "4-Error" defined.
N04.FR.11	When the Charging Station receives a SetVariableMonitoringRequest without an Id AND N04.FR.08	The Charging Station will generate an Id and return it in the SetVariableMonitoringResponse .	
N04.FR.12	When the Charging Station receives a SetVariableMonitoringRequest with an Id AND A monitor exists matching the given Id AND The given Component/Variable combination corresponds with the existing VariableMonitor.	The Charging Station SHALL replace the monitor.	
N04.FR.13	When the Charging Station receives a SetVariableMonitoringRequest with an Id AND No monitor exists matching the given Id.	The Charging Station SHALL set the <i>attributeStatus</i> field in the corresponding SetMonitoringResult to: Rejected .	
N04.FR.14	When the Charging Station receives a SetVariableMonitoringRequest with type Delta and value contains a negative value.	The Charging Station SHALL set the <i>attributeStatus</i> field in the corresponding SetMonitoringResult to: Rejected .	More information can be provided in the optional <i>statusInfo</i> element.
N04.FR.15	N04.FR.12 AND The replaced VariableMonitor belonged to the 'PreconfiguredMonitors'.	The new VariableMonitor shall be classified as a 'CustomMonitor', until reset by a SetMonitoringBaseRequest .	
N04.FR.16	When the Charging Station receives a SetVariableMonitoringRequest with an Id AND a monitor exists matching the given Id AND the given Component/Variable combination does NOT correspond with the existing VariableMonitor.	The Charging Station SHALL respond with Rejected AND NOT replace the VariableMonitor.	It is not allowed to change Variable or Component of a monitor.

ID	Precondition	Requirement definition	Note
N04.FR.17	When the CSMS sends a SetVariableMonitoringRequest with type Delta for a Variable that is NOT of a numeric type	It is RECOMMENDED to use a <i>monitorValue</i> of 1.	<i>monitorValue</i> is irrelevant for non-numeric types (e.g. any type except decimal or integer), since the monitor is triggered by every change of the Variable.
N04.FR.18	N04.FR.12 AND The <i>id</i> in the SetVariableMonitoringRequest refers to a HardWiredMonitor	The Charging Station SHALL respond with Rejected AND NOT replace the VariableMonitor.	It is not possible to change a hardwired monitor.
N04.FR.19	The Charging Station has rebooted	The CSMS IS RECOMMENDED to send a GetMonitoringReportRequest message to get a new list of monitors.	Custom monitors are persistent after reboot or firmware update, but IDs may have changed.

N05 - Set Monitoring Level

Table 222. N05 - Set Monitoring Level

No.	Type	Description
1	Name	Set Monitoring Level
2	ID	N05
	Functional block	N. Diagnostics
3	Objective(s)	To give the CSMS the ability to request the Charging Station to restrict the reporting of monitoring events by NotifyEventRequest to only those monitors with a severity number lower than or equal to a certain severity.
4	Description	It may be desirable to restrict the reporting of monitoring events, to only those monitors with a severity number lower than or equal to a certain severity. For example when the data-traffic between Charging Station and CSMS needs to be limited for some reason. The CSMS can control which events it will be notified of by the Charging Station with the SetMonitoringLevelRequest message.
	Actors	Charging Station, CSMS, CSO
	Scenario description	<ol style="list-style-type: none"> 1. The CSO triggers the CSMS to request a Charging Station to restrict the reporting of monitoring events, by setting a severity level limit. 2. The CSMS sends a SetMonitoringLevelRequest to the Charging Station. 3. The Charging Station responds with a SetMonitoringLevelResponse.
5	Prerequisite(s)	Charging Station supports Monitoring
6	Postcondition(s)	The Charging Station restricted the reporting of monitoring events by NotifyEventRequest to only those wanted by the user.

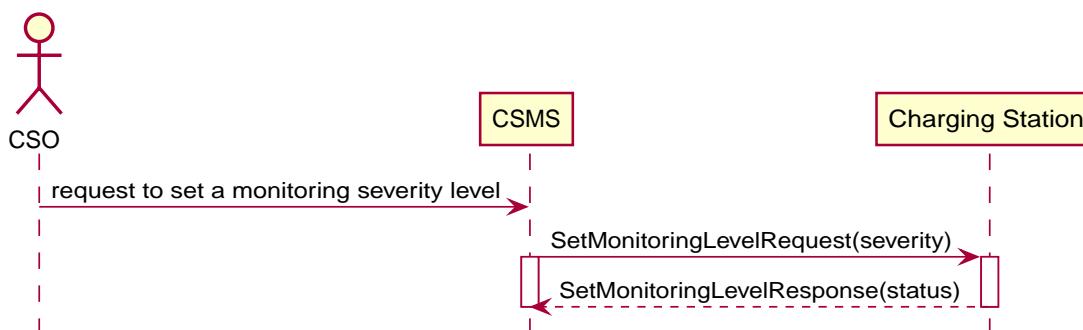


Figure 136. Sequence Diagram: Set Monitoring Level

7	Error handling	n/a
8	Remark(s)	n/a

N05 - Set Monitoring Level - Requirements

Table 223. N05 - Requirements

ID	Precondition	Requirement definition
N05.FR.01	When the Charging Station accepts a <code>setMonitoringLevelRequest</code>	The Charging Station SHALL send a <code>setMonitoringLevelResponse</code> with <code>Accepted</code> .
N05.FR.02	When the Charging Station receives a <code>setMonitoringLevelRequest</code> for a severity that is out of range	The Charging Station SHALL send a <code>setMonitoringLevelResponse</code> with <code>Rejected</code> .
N05.FR.03	N05.FR.01	The Charging Station SHALL restrict the reporting of monitoring events by <code>NotifyEventRequest</code> to only those monitors with a severity number lower than or equal to the given severity.

N06 - Clear / Remove Monitoring

Table 224. N06 - Clear / Remove Monitoring

No.	Type	Description
1	Name	Clear / Remove Monitoring
2	ID	N06
	Functional block	N. Diagnostics
3	Objective(s)	To give the CSMS the ability to clear / remove monitoring settings.
4	Description	A monitoring setting can be cleared (removed) by sending a <code>ClearVariableMonitoringRequest</code> with the id of the monitoring setting.
	Actors	Charging Station, CSMS, CSO
	Scenario description	<ol style="list-style-type: none"> 1. The CSO triggers the CSMS to request clearing/removing one or more variables in a Charging Station. 2. The CSMS sends a <code>ClearVariableMonitoringRequest</code> to the Charging Station. 3. The Charging Station responds with a <code>ClearVariableMonitoringResponse</code>.
5	Prerequisite(s)	Charging Station supports Monitoring
6	Postcondition(s)	The Charging Station cleared / removed the requested monitoring settings.

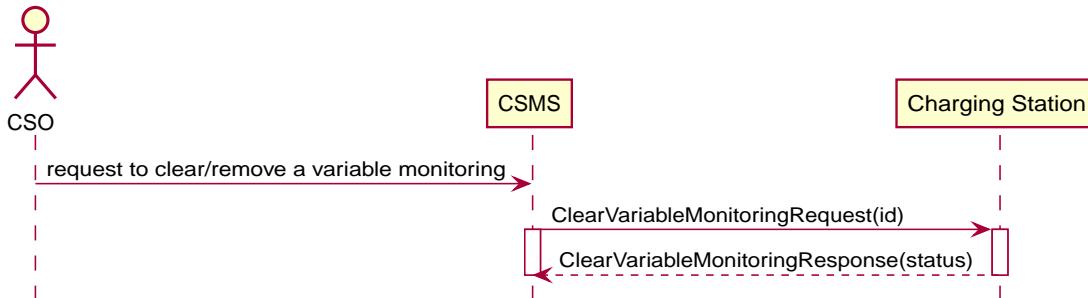


Figure 137. Sequence Diagram: Clear / Remove Monitoring

7	Error handling	n/a
8	Remark(s)	n/a

N06 - Clear / Remove Monitoring - Requirements

Table 225. N06 - Requirements

ID	Precondition	Requirement definition
N06.FR.01	When the Charging Station accepts a <code>ClearVariableMonitoringRequest</code>	The Charging Station SHALL send a <code>ClearVariableMonitoringResponse</code> with <code>Accepted</code> .
N06.FR.02	When the Charging Station receives a <code>ClearVariableMonitoringRequest</code> with a non existing id	The Charging Station SHALL send a <code>ClearVariableMonitoringResponse</code> with <code>NotFound</code> .

ID	Precondition	Requirement definition
N06.FR.03	When the Charging Station receives a ClearVariableMonitoringRequest for an <i>id</i> referring to a monitor that cannot be cleared (for example because it is hardcoded).	The Charging Station SHALL send a ClearVariableMonitoringResponse with Rejected .
N06.FR.04		The CSMS SHALL NOT put more <i>id</i> elements in a ClearVariableMonitoringRequest than reported by the Charging Station via: ItemsPerMessageClearVariableMonitoring and BytesPerMessageClearVariableMonitoring .
N06.FR.05		For every <i>id</i> in a ClearVariableMonitoringRequest the Charging Station SHALL add a clearMonitoringResult element to the ClearVariableMonitoringResponse sent to the CSMS.
N06.FR.06	Charging Station receives a ClearVariableMonitoringRequest with more <i>id</i> elements than allowed by ItemsPerMessageClearVariableMonitoring	The Charging Station MAY respond with a CALLERROR(OccurrenceConstraintViolation)
N06.FR.07	Charging Station receives a ClearVariableMonitoringRequest with a length of more bytes than allowed by BytesPerMessageClearVariableMonitoring	The Charging Station MAY respond with a CALLERROR(FormatViolation)

2.3. Monitoring Events

N07 - Alert Event

Table 226. N07 - Alert Event

No.	Type	Description
1	Name	Alert Event
2	ID	N07
	Functional block	N. Diagnostics
3	Objective(s)	To give the Charging Station the ability to notify the CSMS about monitoring events.
4	Description	NotifyEventRequest reports every Component/Variable for which a VariableMonitoring setting was triggered. Only the VariableMonitoring settings that are responsible for triggering an event are included.
	Actors	Charging Station, CSMS
	Scenario description	1. If a threshold or a delta value has exceeded, the Charging Station sends a NotifyEventRequest to the CSMS. 2. The CSMS responds with a NotifyEventResponse .
5	Prerequisite(s)	The Charging Station has active monitoring settings. The monitoring setting(s) might have been configured explicitly via a SetVariableMonitoring message or it might be "hard-wired" in the Charging Station's firmware.
6	Postcondition(s)	The Charging Station notified the CSMS about the monitoring events.

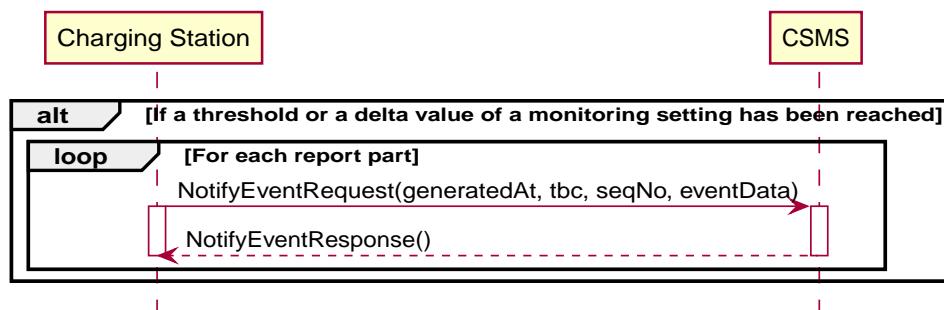


Figure 138. Sequence Diagram: Alert Event

7	Error handling	n/a
---	----------------	-----

8	Remark(s)	<p>Requirement N07.FR.04 states that events with a severity equal or less than OfflineMonitoringEventQueuingSeverity shall be queued while the charging station is offline, and delivered once online. This implies that events with a severity greater than OfflineMonitoringEventQueuingSeverity will not be sent to CSMS. The result is, that the logical chain of events may be broken when the charging station is back online.</p> <p>For example, a monitoring event for a variable exceeding a threshold occurred while offline and was not sent. Once back online, at some point in time the monitoring event is reported with the variable <i>cleared</i> set to true, but CSMS did not even know that the threshold had been exceeded. CSMS will have to be able to deal with that.</p> <p>This problem can be prevented, while still adhering to the specification, by not simply discarding these monitoring events, but by delaying the evaluation of those monitors that exceed OfflineMonitoringEventQueuingSeverity, until the charging station comes back online. The result is, that when the charging station is back online, CSMS will get the monitoring events that apply to the current situation, and it is fully up-to-date regarding the monitors. Only those monitoring events that were triggered & cleared during the offline period will remain invisible to CSMS.</p>
---	------------------	---

N07 - Alert Event - Requirements

Table 227. N07 - Requirements

ID	Precondition	Requirement definition	Note
N07.FR.02	When a monitored value returns to within the set <i>UpperThreshold</i> or <i>LowerThreshold</i>	The Charging Station SHALL send a NotifyEventRequest with an eventData with the attribute <i>cleared</i> is true.	
N07.FR.03	When the CSMS receives an notifyEventRequest	The CSMS SHALL respond with an empty NotifyEventResponse .	
N07.FR.04	When a monitor is triggered AND The severity number of the monitor is equal to or lower than the severity number set in the Configuration Variable OfflineMonitoringEventQueueingSeverity AND The Charging Station is <i>offline</i>	The Charging Station SHALL queue this NotifyEventRequest and deliver it when it is back online.	
N07.FR.05	When a monitor is triggered AND another event caused this event	The Charging Station MAY include the <i>eventId</i> of the other event in the <i>cause</i> field of the eventData element in the NotifyEventRequest message.	
N07.FR.06	When a monitor is triggered	An eventData element in a NotifyEventRequest SHALL contain the Component , Variable and variableMonitoringId that caused the event.	
N07.FR.07	When a monitor is triggered	The Charging Station SHALL set the <i>seqNo</i> of the first NotifyEventRequest sent for this event to 0.	
N07.FR.10	When a monitor is triggered AND A variableMonitoring setting has been set on a write-only variable.	The <i>actualField</i> of the NotifyEventRequest SHALL be empty.	
N07.FR.11	When modifying a set <i>UpperThreshold</i> or <i>LowerThreshold</i> VariableMonitor.	The Charging Station SHALL check if the new threshold clears the old threshold OR if the new threshold is exceeded by the monitored value.	
N07.FR.12	When removing a set <i>UpperThreshold</i> or <i>LowerThreshold</i> VariableMonitor AND the threshold was exceeded.	The Charging Station SHALL NOT send a NotifyEventRequest with an eventData with the attribute <i>cleared</i> is true.	
N07.FR.13		A VariableMonitoring needs to be stored persistently across reboots.	
N07.FR.14	When a variableMonitoring setting of type <i>UpperThreshold</i> or <i>LowerThreshold</i> has been triggered AND after a reboot occurred the monitored value returned within the configured threshold.	The Charging Station SHALL send a NotifyEventRequest with an eventData with the attribute <i>cleared</i> is true.	

ID	Precondition	Requirement definition	Note
N07.FR.15	When a monitor is triggered AND The severity of the monitor is greater than the monitoring severity level set in a SetMonitoringLevelRequest by the CSMS (see use case N05 - Set Monitoring Level)	The Charging Station SHALL NOT send a NotifyEventRequest for the triggered monitor.	
N07.FR.16	When there is a monitor with type UpperThreshold on a Component/Variable combination AND the Actual value (attributeType Actual) of the Variable exceeds <i>monitorValue</i>	The Charging Station SHALL send a NotifyEventRequest with trigger Alerting for the triggered monitor.	Notification is sent when exceeding the threshold, not on the threshold.
N07.FR.17	When there is a monitor with type LowerThreshold on a Component/Variable combination AND the Actual value (attributeType Actual) of the Variable drops below <i>monitorValue</i>	The Charging Station SHALL send a NotifyEventRequest with trigger Alerting for the triggered monitor.	Notification is sent when dropping below the threshold, not on the threshold.
N07.FR.18	When there is a monitor with type Delta on a Component/Variable combination AND the Variable is of a numeric type AND the Actual value (attributeType Actual) of the Variable has changed more than plus or minus <i>monitorValue</i> since the time that this monitor was set or since the last time this event notice was sent, whichever was last	The Charging Station SHALL send a NotifyEventRequest with trigger Delta for the triggered monitor.	
N07.FR.19	When there is a monitor with type Delta on a Component/Variable combination AND the Variable is NOT of a numeric type AND the Actual value (attributeType Actual) of the Variable has changed since the time that this monitor was set or since the last time this event notice was sent, whichever was last (Note: For variables that are not numeric, like boolean, string or enumerations, a monitor of type Delta will trigger an event notice whenever the variable changes, regardless of the value of <i>monitorValue</i>)	The Charging Station SHALL send a NotifyEventRequest with trigger Delta for the triggered monitor.	

N08 - Periodic Event

Table 228. N08 - Periodic Event

No.	Type	Description
1	Name	Periodic Event
2	ID	N08
	Functional block	N. Diagnostics
3	Objective(s)	To give the Charging Station the ability to notify the CSMS periodically about monitoring events.
4	Description	NotifyEventRequest reports every Component/Variable for which a VariableMonitoring setting was triggered. Only the VariableMonitoring settings that are responsible for triggering an event are included.
	Actors	Charging Station, CSMS

No.	Type	Description
	Scenario description	1. If a periodic value has exceeded, the Charging Station sends a NotifyEventRequest with trigger periodic to the CSMS. 2. The CSMS responds with a NotifyEventResponse .
5	Prerequisite(s)	The Charging Station has active monitoring settings. The monitoring setting(s) might have been configured explicitly via a SetVariableMonitoring message or it might be "hard-wired" in the Charging Station's firmware.
6	Postcondition(s)	The Charging Station notified the CSMS about the monitoring events.

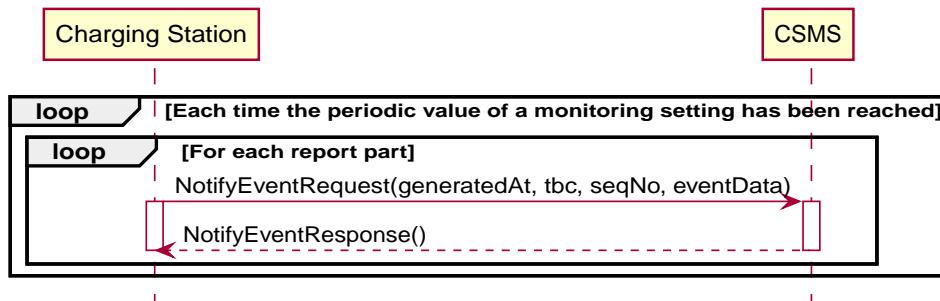


Figure 139. Sequence Diagram: Periodic Event

7	Error handling	n/a
8	Remark(s)	n/a

N08 - Periodic Event - Requirements

Table 229. N08 - Requirements

ID	Precondition	Requirement definition
N08.FR.02	When the CSMS receives an NotifyEventRequest	The CSMS SHALL respond with an empty NotifyEventResponse .
N08.FR.03	N08.FR.06 OR N08.FR.07 AND The severity number of the monitor is equal to or lower than the severity number set in the Configuration Variable OfflineMonitoringEventQueueingSeverity AND The Charging Station is <i>offline</i>	The Charging Station SHALL queue this NotifyEventRequest and deliver it when it is back online.
N08.FR.04	N08.FR.06 OR N08.FR.07 AND This NotifyEventRequest is the first or only report part.	The Charging Station SHALL set <code>seqNo</code> to 0.
N08.FR.05	N08.FR.06 OR N08.FR.07 AND When the variableMonitoring setting which triggered the event is either of type Periodic or PeriodicClockAligned	The Charging Station SHALL set <code>trigger</code> to Periodic .
N08.FR.06	When there is a monitor with type Periodic on a Component/Variable combination AND the number of seconds specified in <code>monitorValue</code> have passed (starting from the time that this monitor was set or triggered)	The Charging Station SHALL send a NotifyEventRequest with trigger Periodic for the triggered monitor.
N08.FR.07	When there is a monitor with type PeriodicClockAligned on a Component/Variable combination AND the number of seconds specified by <code>monitorValue</code> , starting from the nearest clock-aligned interval after this monitor was set, have passed (For example, a <code>monitorValue</code> of 900 will trigger event notices at 0, 15, 30 and 45 minutes after the hour, every hour)	The Charging Station SHALL send a NotifyEventRequest with trigger Periodic for the triggered monitor.

2.4. Customer Information

N09 - Get Customer Information

Table 230. N09 - Get Customer Information

No.	Type	Description
1	Name	Get Customer Information
2	ID	N09
	Functional block	N. Diagnostics
3	Objective(s)	To enable the CSMS to retrieve raw customer information from a Charging Station.
4	Description	The CSMS sends a message to the Charging Station to retrieve raw customer information, for example to be compliant with local privacy laws. The Charging Station notifies the CSMS by sending one or more reports.
	Actors	Charging Station, CSMS
	Scenario description	<p>1. The CSMS sends a CustomerInformationRequest with the report flag set to <i>true</i> to the Charging Station with a reference to a customer (<i>idToken</i>, <i>customerCertificate</i> or <i>customerIdentifier</i>).</p> <p>2. The Charging Station responds with CustomerInformationResponse, indicating whether it will send it or not.</p> <p>3. The Charging Station sends one or more NotifyCustomerInformationRequest messages to the CSMS.</p> <p>4. The CSMS responds with one or more NotifyCustomerInformationResponse messages to the Charging Station.</p>
5	Prerequisite(s)	n/a
6	Postcondition(s)	The CSMS has <i>Successfully received</i> a CustomerInformationResponse message with status <i>Accepted</i> AND has <i>Successfully received</i> the requested data.

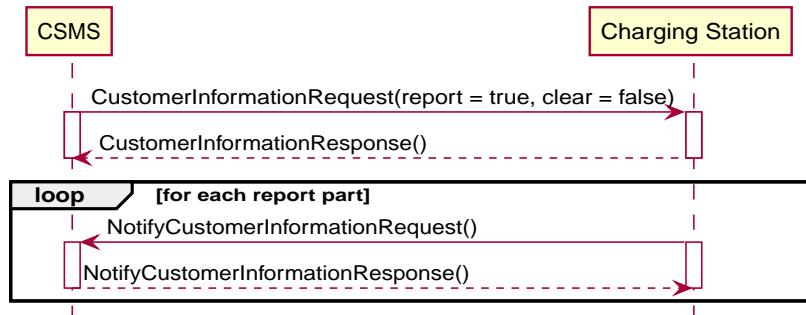


Figure 140. Sequence Diagram: Get Customer Information

7	Error handling	n/a
8	Remark(s)	n/a

N09 - Get Customer Information - Requirements

Table 231. N09 - Requirements

ID	Precondition	Requirement definition	Note
N09.FR.01	When the CSMS wants to retrieve CustomerInformation from the Charging Station.	The report flag in the CustomerInformationRequest SHALL be set to <i>true</i> .	
N09.FR.02	When the Charging Station receives a CustomerInformationRequest AND it is in a state where it can process this request.	the Charging Station SHALL respond with a CustomerInformationResponse message with status <i>Accepted</i> .	
N09.FR.03	When the Charging Station is in a state where it cannot process this request.	On receipt of the CustomerInformationRequest the Charging Station SHALL respond with a CustomerInformationResponse with status <i>Rejected</i> .	
N09.FR.04		The CSMS SHALL include a reference to a customer by including either an idToken , customerCertificate or customerIdentifier in the CustomerInformationRequest .	
N09.FR.05	N09.FR.02 AND the Charging Station has information stored about the customer referred to by the customer identifier	The Charging Station SHALL send the requested information via one or more NotifyCustomerInformationRequest messages to the CSMS.	
N09.FR.06	N09.FR.02 AND the Charging Station has no information stored about the customer referred to by the customer identifier.	The Charging Station SHALL send one NotifyCustomerInformationRequest message to the CSMS indicating that no data was found.	
N09.FR.07	When receiving a CustomerInformationRequest with both the report flag as well as the clear flag are set to <i>false</i>	It is RECOMMENDED to respond with status a CustomerInformationResponse message with status <i>Rejected</i> .	
N09.FR.08	When requesting user information according to the customerCertificate	The CSMS SHALL use the hashAlgorithm , which was used to install the certificate.	When a new firmware is installed it is RECOMMENDED that the CSMS requests the certificate first using GetInstalledCertificatesId s Request to be sure of the used hashAlgorithm .

N10 - Clear Customer Information

Table 232. N10 - Clear Customer Information

No.	Type	Description
1	Name	Clear Customer Information
2	ID	N10
	<i>Functional block</i>	N. Diagnostics
3	Objective(s)	To enable the CSMS to clear (and retrieve) raw customer information from a Charging Station.
4	Description	The CSMS sends a message to the Charging Station to clear (and retrieve) raw customer information, for example to be compliant with local privacy laws. The Charging Station notifies the CSMS by sending one or more reports.
	Actors	Charging Station, CSMS

No.	Type	Description
	Scenario description	<p>1. The CSMS sends CustomerInformationRequest with the clear flag set to <i>true</i> to the Charging Station with a reference to a customer (<i>idToken</i>, <i>customerCertificate</i> or <i>customerIdentifier</i>).</p> <p>2. The Charging Station responds with CustomerInformationResponse, indicating whether it will send it or not.</p> <p>3. If the report flag is set to <i>true</i>, the Charging Station sends one or more NotifyCustomerInformationRequest messages to the CSMS.</p> <p>4. The CSMS responds with one or more NotifyCustomerInformationResponse messages to the Charging Station.</p>
5	Prerequisite(s)	n/a
6	Postcondition(s)	The CSMS has <i>Successfully received</i> a CustomerInformationResponse message with status <i>Accepted</i> , the Charging Station has removed the customer information as requested and (if report flag was set to <i>true</i>) the CSMS has <i>Successfully received</i> the removed data.

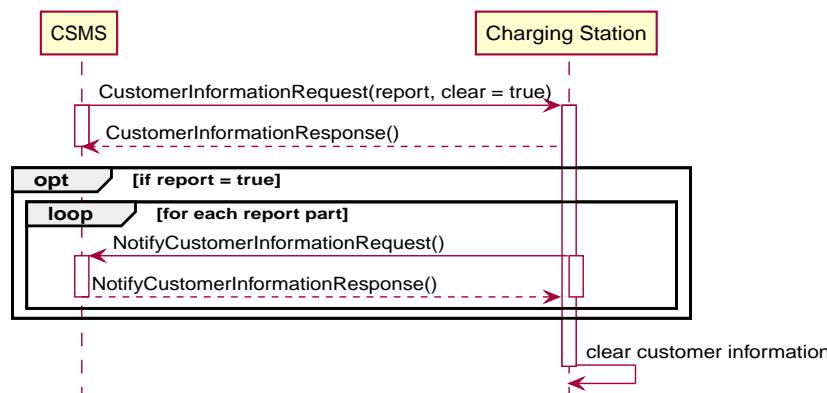


Figure 141. Sequence Diagram: Clear Customer Information

7	Error handling	n/a
8	Remark(s)	n/a

N10 - Clear Customer Information - Requirements

Table 233. N10 - Requirements

ID	Precondition	Requirement definition	Note
N10.FR.01	When the Charging Station receives a CustomerInformationRequest AND it is in a state where it can process this request.	the Charging Station SHALL respond with a CustomerInformationResponse message with status <i>Accepted</i> .	
N10.FR.02	When the Customer referred to by the customer identifier is present in the Local Authorization List of a Charging Station	The CSMS SHALL update the Local Authorization List using the SendLocalListRequest (see D01 - Send Local Authorization List).	To prevent problems with Local Authorization List versions.
N10.FR.03	N10.FR.01 AND receiving a CustomerInformationRequest with the clear flag set to <i>true</i> and the report flag set to <i>true</i> AND the Charging Station has information stored about the customer referred to by the customer identifier.	The Charging Station SHALL remove all customer related data for the Customer referred to by the customer identifier from the Charging Station, except from the Locallist AND the Charging Station SHALL send the cleared information via one or more NotifyCustomerInformationRequest messages to the CSMS.	To prevent problems with Locallist versions only the CSMS can change the contents of the Locallist.
N10.FR.04	N10.FR.01 AND receiving a CustomerInformationRequest with the clear flag set to <i>true</i> and the report flag set to <i>true</i> AND the Charging Station has no information stored about the customer referred to by the customer identifier.	The Charging Station SHALL send one NotifyCustomerInformationRequest message to the CSMS indicating that no data was found.	

ID	Precondition	Requirement definition	Note
N10.FR.05	When the Charging Station receives a CustomerInformationRequest and is in a state where it cannot process this request.	The Charging Station SHALL respond with a CustomerInformationResponse with status <i>Rejected</i>	
N10.FR.06	N10.FR.01 AND receiving a CustomerInformationRequest with the clear flag set to <i>true</i> , the report flag set to <i>false</i>	The Charging Station SHALL remove all customer related data for the Customer referred to by the customer identifier from the Charging Station, except from the LocalList AND the Charging Station SHALL send one NotifyCustomerInformationRequest message to the CSMS indicating that the data was cleared.	To prevent problems with LocalList versions only the CSMS can change the contents of the LocalList.
N10.FR.07	When receiving a CustomerInformationRequest with both the report flag as well as the clear flag are set to <i>false</i>	It is RECOMMENDED to respond with a CustomerInformationResponse message with status <i>Rejected</i> .	
N10.FR.08		The CSMS SHALL include a reference to a customer by including either an idToken , customerCertificate or customerIdentifier in the CustomerInformationRequest .	
N10.FR.09	When clearing user information according to the customerCertificate	The CSMS SHALL use the <i>hashAlgorithm</i> , which was used to install the certificate.	When a new firmware is installed it is RECOMMENDED that the CSMS requests the certificate first using GetInstalledCertificatesRequest to be sure of the used <i>hashAlgorithm</i> .

O. DisplayMessage

1. Introduction

With the DisplayMessage feature, OCPP enables a CSO to display a message or a cycle of messages on a Charging Station, that is not part of the firmware of the Charging Station. The CSO gets control over these messages: the CSO can set, retrieve (get), replace and clear messages.

Every message can be configured in different languages and different message formats. See [DisplayMessageSupportedFormats](#). So the Charging Station can select the correct format/language when it needs to display a message to a user. Every message the CSO sends to the Charging Station has some parameters to control when and how a message is shown: priority, state, start/end time etc. See [DisplayMessageSupportedPriorities](#).

NOTE

It is not possible to retrieve/modify messages not configured via SetDisplayMessageRequest. (In other words: Message coded in the firmware of a Charging Station cannot be modified.)

2. Use cases & Requirements

001 - Set DisplayMessage

Table 234. 001 - Set DisplayMessage

No.	Type	Description
1	Name	Set DisplayMessage
2	ID	001
	Functional block	O. DisplayMessage
3	Objectives	To enable a CSO to display additional messages on a Charging Station that are not part of the firmware.
4	Description	This use case describes how a CSO can set a message to be displayed on a Charging Station. Depending on the given parameters the message shall be displayed a certain way and at a certain moment on the Charging Station.
	Actors	CSO, CSMS, Charging Station
	Scenario description	<ol style="list-style-type: none"> 1. The CSO configures the CSMS to send a request to set a new message. 2. The CSMS sends a SetDisplayMessageRequest message to the Charging Station. 3. The Charging Station accepts the request by sending a SetDisplayMessageResponse message to the CSMS. 4. The Charging Station shows the new message on the display at the configured moment.
	Alternative scenario's	002 - Set DisplayMessage for Transaction 006 - Replace DisplayMessage
5	Prerequisites	No messages configured with the same IDs.
6	Postcondition(s)	The new message will be displayed on the Charging Station (time, duration and position depending on configuration)

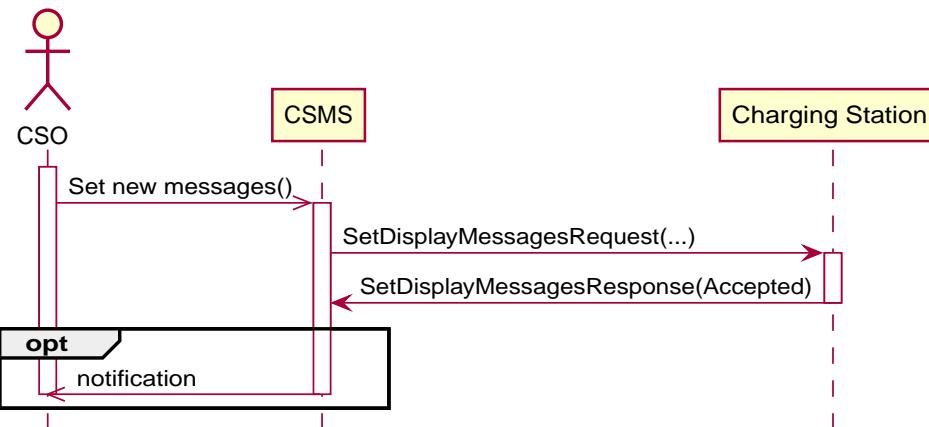


Figure 142. Set DisplayMessage sequence diagram

7	Error Handling	n/a
8	Remarks	The maximum number of messages that can be stored in a Charging Station can be read by the CSMS in the Configuration Variable: NumberOfDisplayMessages.maxLimit .

001 - Set DisplayMessage - Requirements

Table 235. 001 - Set DisplayMessage - Requirements

ID	Precondition	Requirement definition
001.FR.01	When the Charging Station receives a MessageInfo object via a SetDisplayMessageRequest and the priority of the message is not supported by the Charging Station	The Charging Station SHALL send a SetDisplayMessageResponse with status: NotSupportedPriority .

ID	Precondition	Requirement definition
001.FR.02	When the Charging Station receives a MessageInfo object via a SetDisplayMessageRequest and the state of the message is not supported by the Charging Station	The Charging Station SHALL send a SetDisplayMessageResponse with status: NotSupportedState .
001.FR.03	When the Charging Station receives a MessageInfo object via a SetDisplayMessageRequest and the format of the message is not supported by the Charging Station	The Charging Station SHALL send a SetDisplayMessageResponse with status: NotSupportedMessageFormat .
001.FR.04		When a CSMS sends a message to a Charging Station that does not belong to a transaction, the field: transactionId in the Message field SHALL be omitted.
001.FR.05		The CSMS MAY include a startTime and endTime when setting a message.
001.FR.06	001.FR.05	The Charging Station SHALL NOT display the DisplayMessage message before the startTime .
001.FR.07	001.FR.05	The Charging Station SHALL remove a DisplayMessage message after the endTime .
001.FR.08	When the Charging Station knows the language preferences of the EV Driver	The Charging Station SHALL display the DisplayMessage message in the preferred language, if available.
001.FR.09	001.FR.08	When no matching language is available, it is RECOMMENDED to show a DisplayMessage message in English as fall-back, if available.
001.FR.10		The Charging Station SHALL store the messages in persistent storage, so they survive a power cycle/reboot of the Charging Station.
001.FR.11	When the Charging Station receives a SetDisplayMessageRequest and the total number of messages after having handled this request will exceed NumberOfDisplayMessages.maxLimit .	The Charging Station SHALL respond with status: Rejected .
001.FR.12	When the Charging Station receives a SetDisplayMessageRequest and the priority of the message is NormalCycle	The Charging Station SHALL show this message at the configured moment in the normal cycle of messages.
001.FR.13	When the Charging Station receives a SetDisplayMessageRequest and the priority of the message is InFront	The Charging Station SHALL show this message at the configured moment, regardless of the normal cycle of messages.
001.FR.14	When multiple messages with priority InFront are configured to be shown at the same time	The Charging Station SHALL cycle these messages.
001.FR.15	When the Charging Station receives a SetDisplayMessageRequest and the priority of the message is AlwaysFront	The Charging Station SHALL show this message at the configured moment, regardless of other installed messages. Hence, it shall not cycle it with other messages and the Charging Station's own messages shall not override this message.
001.FR.16	001.FR.15 AND Another message with priority AlwaysFront is already set	The Charging Station SHALL replace the old message with the newly set message.
001.FR.17		Language SHALL be specified as RFC-5646 tags, see: [RFC5646] , example: US English is: "en-US"

002 - Set DisplayMessage for Transaction

Table 236. 002 - Set DisplayMessage for Transaction

No.	Type	Description
1	Name	Set DisplayMessage for Transaction
2	ID	002
	Functional block	O. DisplayMessage
	Parent use case	001 - Set DisplayMessage
3	Objectives	To enable a CSO to display messages during an ongoing transaction on a Charging Station that are not build in to the firmware.
4	Description	This use case describes how a CSO can set a message to be displayed on a Charging Station for a specific transaction. Depending on the given parameters the message shall be displayed a certain way on the Charging Station.
	Actors	CSO, CSMS, Charging Station
	Scenario description	<ol style="list-style-type: none"> 1. The CSO configures the CSMS to send a request to show a new message during a given transaction. 2. The CSMS sends a SetDisplayMessageRequest message with the transactionId to the Charging Station. 3. The Charging Station accepts the request by sending a SetDisplayMessageResponse message to the CSMS. 4. The Charging Station shows the new message on the display while the transaction is ongoing.
	Alternative scenario's	001 - Set MessageMessage 006 - Replace MessageMessage
5	Prerequisites	No messages configured with the same IDs.
6	Postcondition(s)	The new message will be displayed on the Charging Station while the transaction is ongoing (time, duration and position depend on configuration)

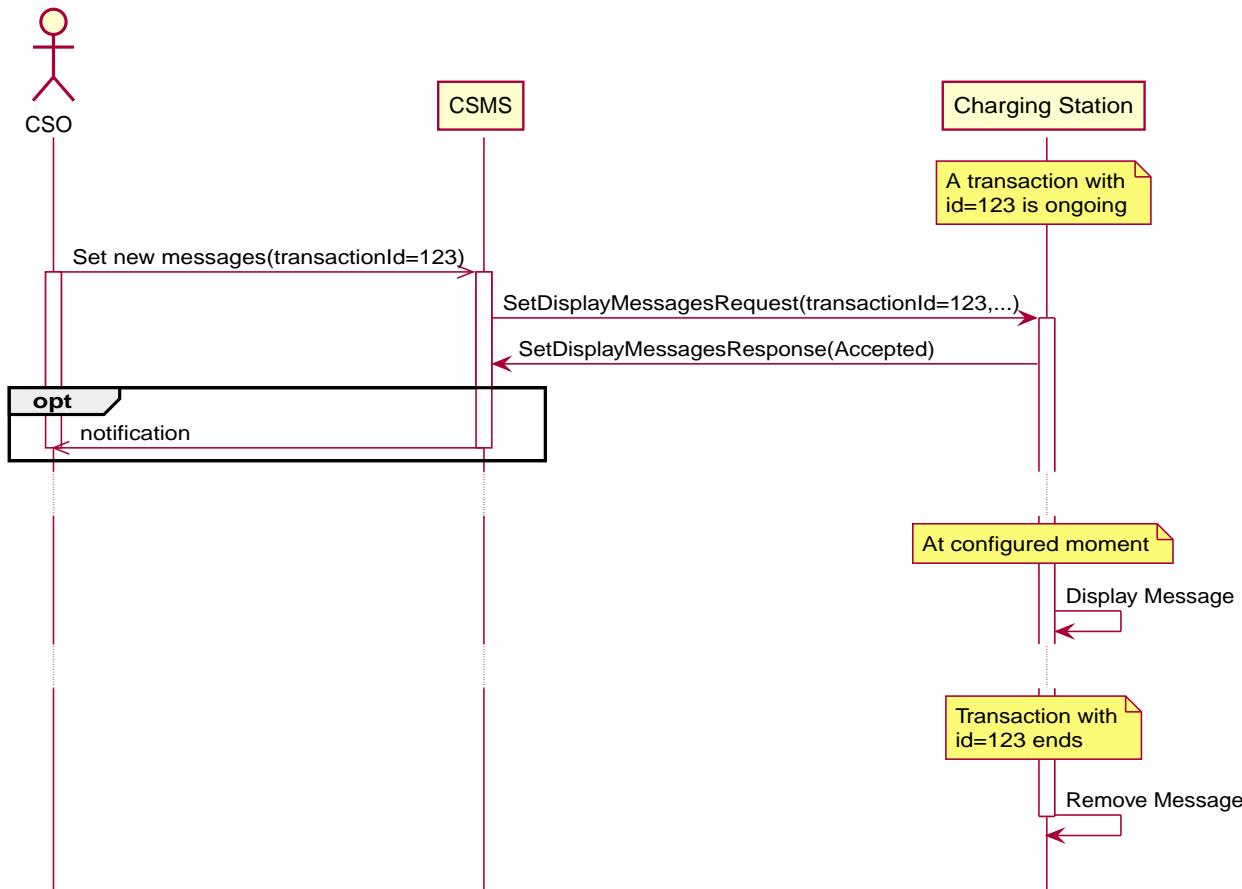


Figure 143. Set DisplayMessage for transaction sequence diagram

7	Error Handling	n/a
---	----------------	-----

8	Remarks	The maximum number of messages that can be stored in a Charging Station can be read by the CSMS in the Configuration Variable: NumberOfDisplayMessages.maxLimit .
---	----------------	---

002 - Set DisplayMessage for Transaction - Requirements

Table 237. 002 - Set DisplayMessage for Transaction - Requirements

ID	Precondition	Requirement definition
002.FR.01	When the Charging Station receives a Message object via a SetDisplayMessageRequest and the transactionId of the message is not known by the Charging Station	The Charging Station SHALL send a SetDisplayMessageResponse with status: UnknownTransaction .
002.FR.02	When the transaction with the given transactionId ends	The Charging Station SHALL remove the message from the list of messages.
002.FR.03	When the Charging Station receives a MessageInfo object via a SetDisplayMessageRequest and the priority of the message is not supported by the Charging Station	The Charging Station SHALL send a SetDisplayMessageResponse with status: NotSupportedPriority .
002.FR.04	When the Charging Station receives a MessageInfo object via a SetDisplayMessageRequest and the state of the message is not supported by the Charging Station	The Charging Station SHALL send a SetDisplayMessageResponse with status: NotSupportedState .
002.FR.05	When the Charging Station receives a MessageInfo object via a SetDisplayMessageRequest and the format of the message is not supported by the Charging Station	The Charging Station SHALL send a SetDisplayMessageResponse with status: NotSupportedMessageFormat .
002.FR.06		The Charging Station SHALL NOT display the DisplayMessage message before the startTime .
002.FR.07		The Charging Station SHALL remove a DisplayMessage message after the endTime .
002.FR.08	When the Charging Station knows the language preferences of the EV Driver	The Charging Station SHALL display the DisplayMessage message in the preferred language, if available.
002.FR.09	002.FR.08	When no matching language is available, it is RECOMMENDED to show a DisplayMessage message in English as fall-back, if available.
002.FR.10		The Charging Station SHALL store the messages in persistent storage, so they survive a power cycle/reboot of the Charging Station.
002.FR.11	When the Charging Station receives a SetDisplayMessageRequest and the total number of messages after having handled this request will exceed NumberOfDisplayMessages.maxLimit .	The Charging Station SHALL respond with status: Rejected .
002.FR.12		Language SHALL be specified as RFC-5646 tags, see: [RFC5646] , example: US English is: "en-US"
002.FR.14	When the Charging Station receives a SetDisplayMessageRequest and the priority of the message is NormalCycle	The Charging Station SHALL show this message in the normal cycle of messages.
002.FR.15	When the Charging Station receives a SetDisplayMessageRequest and the priority of the message is InFront	The Charging Station SHALL show this message at the configured moment, regardless of the normal cycle of messages.
002.FR.16	When multiple messages with priority InFront are configured to be shown at the same time	The Charging Station SHALL cycle these messages.
002.FR.17	When the Charging Station receives a SetDisplayMessageRequest and the priority of the message is AlwaysFront	The Charging Station SHALL show this message at the configured moment, regardless of other installed messages. Hence, it shall not cycle it with other messages and the Charging Station's own message shall not override this message.

ID	Precondition	Requirement definition
002.FR.18	002.FR.17 AND Another message with priority <i>AlwaysFront</i> is already set	The Charging Station SHALL replace the old message with the newly set message.

003 - Get All DisplayMessages

Table 238. 003 - Get All DisplayMessage IDs

No.	Type	Description
1	Name	Get All DisplayMessages
2	ID	003
	Functional block	O. DisplayMessage
3	Objectives	Enable a CSO to retrieve all messages currently configured in a Charging Station.
4	Description	<p>This use case describes how a CSO can request all the installed DisplayMessages configured via OCPP in a Charging Station.</p> <p>The Charging Station can remove messages when they are out-dated, or transactions have ended. It can be very useful for a CSO to be able to view the current list of messages, so the CSO knows which messages are (still) configured.</p>
	Actors	CSO, CSMS, Charging Station
	Scenario description	<ol style="list-style-type: none"> 1. The CSO asks the CSMS to retrieve all messages. 2. The CSMS sends a GetDisplayMessagesRequest message to the Charging Station. 3. The Charging Station responds with a GetDisplayMessagesResponse Accepted, indicating it has configured messages and will send them. 4. The Charging Station sends one or more NotifyDisplayMessagesRequest messages to the CSMS (depending on the amount of messages to be sent). 5. The CSMS responds to every notify with a NotifyDisplayMessagesResponse message.
5	Prerequisites	There is at least one message configured in the Charging Station
6	Postcondition(s)	n/a

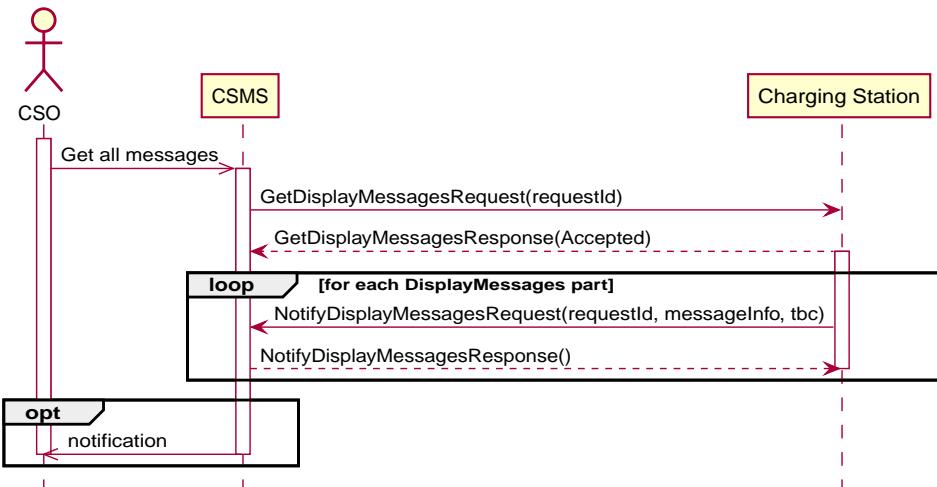


Figure 144. Get All DisplayMessages sequence diagram

7	Error Handling	n/a
8	Remarks	Only messages configured via OCPP can be retrieved via a GetDisplayMessagesRequest .

003 - Get All DisplayMessages - Requirements

Table 239. 003 - Get All DisplayMessage IDs - Requirements

ID	Precondition	Requirement definition
003.FR.01	When all fields except <code>requestId</code> in a GetDisplayMessagesRequest are omitted AND at least one display message is configured.	The Charging Station SHALL respond with Accepted.

ID	Precondition	Requirement definition
003.FR.02	003.FR.01	The Charging Station SHALL send all configured DisplayMessages via NotifyDisplayMessagesRequest .
003.FR.03	003.FR.02 AND There are more DisplayMessages than the Charging Station can send in 1 NotifyDisplayMessagesRequest	The Charging Station SHALL split the DisplayMessages over multiple NotifyDisplayMessagesRequest messages.
003.FR.04	003.FR.03	The Charging Station SHALL set the <i>tbc</i> field is <i>true</i> in every NotifyDisplayMessagesRequest messages, except the last.
003.FR.05	003.FR.04	The Charging Station SHALL set the <i>requestId</i> field to the same value as the <i>requestId</i> in the GetDisplayMessagesRequest .
003.FR.06	When NO DisplayMessages are configured	The Charging Station SHALL respond with <i>Unknown</i> .

004 - Get Specific DisplayMessages

Table 240. 004 - Get a Specific DisplayMessages

No.	Type	Description
1	Name	Get Specific DisplayMessages
2	ID	004
	<i>Functional block</i>	O. DisplayMessage
3	Objectives	Enable a CSO to retrieve one or more specific DisplayMessages, currently configured in a Charging Station.
4	Description	This use case describes how a CSO can request/query for (specific) DisplayMessage, configured via OCPP in a Charging Station. The Charging Station can remove messages when they are outdated, or transactions have ended. It can be very useful for a CSO to be able query the Charging Station for installed DisplayMessages, so the CSO known which messages are (still) configured.
	Actors	CSO, CSMS, Charging Station
	Scenario description	<ol style="list-style-type: none"> 1. The CSO asks the CSMS to query for DisplayMessages. 2. The CSMS sends a GetDisplayMessagesRequest message with the query parameters to the Charging Station. 3. When the Charging Station has DisplayMessages that match the requested parameters, it responds with GetDisplayMessagesResponse Accepted. 4. The Charging Station sends one or more NotifyDisplayMessagesRequest message to the CSMS (depending on the amount of messages to be send). 5. The CSMS response every notify with a NotifyDisplayMessagesResponse message.
5	Prerequisites	There is a message with the given id configured in the Charging Station
6	Postcondition(s)	n/a

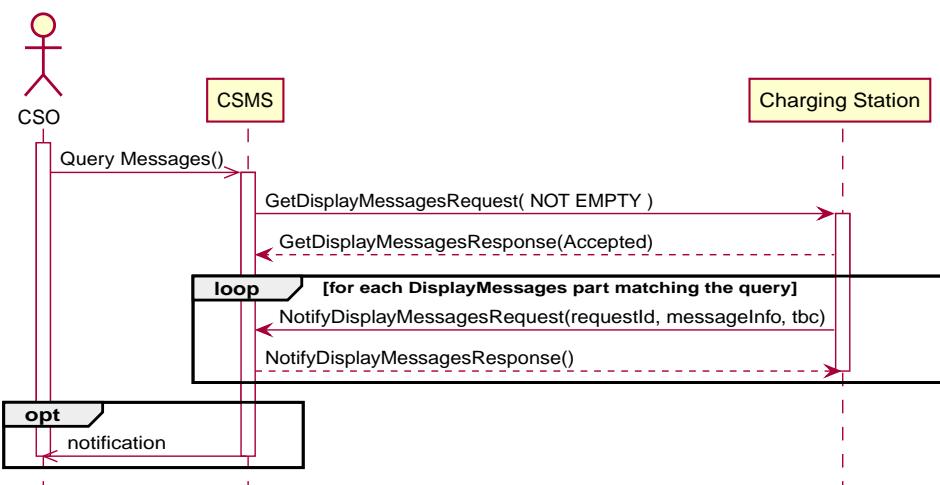


Figure 145. Get a specific *DisplayMessages* sequence diagram

7	Error Handling	n/a
8	Remarks	Only message configured via OCPP can be retrieved via GetDisplayMessagesRequest .

004 - Get Specific DisplayMessage - Requirements

Table 241. 004 - Get Specific DisplayMessages - Requirements

ID	Precondition	Requirement definition
004.FR.01	<p>When one or more of the fields in a GetDisplayMessagesRequest are used AND The Charging Station has DisplayMessages configured that match the parameters in the request</p>	The Charging Station SHALL respond with Accepted.

ID	Precondition	Requirement definition
O04.FR.02	When one or more of the fields in a GetDisplayMessagesRequest are used AND The Charging Station has NO DisplayMessages configured that match the parameters in the request	The Charging Station SHALL respond with <i>Unknown</i> .
O04.FR.03	O04.FR.01	The Charging Station SHALL send all configured DisplayMessages via NotifyDisplayMessagesRequest .
O04.FR.04	O04.FR.03 AND There are more DisplayMessages than the Charging Station can send in 1 NotifyDisplayMessagesRequest	The Charging Station SHALL split the DisplayMessages over multiple NotifyDisplayMessagesRequest messages.
O04.FR.05	O04.FR.04	The Charging Station SHALL set the <i>tbc</i> field is <i>true</i> in every NotifyDisplayMessagesRequest messages, except the last.
O04.FR.06	O04.FR.05	The Charging Station SHALL set the <i>requestId</i> field to the same value as the <i>requestId</i> in the GetDisplayMessagesRequest .
O04.FR.07	When NO DisplayMessages are configured	The Charging Station SHALL respond with <i>Unknown</i> .

005 - Clear a DisplayMessage

Table 242. 005 - Clear a DisplayMessage

No.	Type	Description
1	Name	Clear a DisplayMessage
2	ID	005
	Functional block	O. DisplayMessage
3	Objectives	Enable a CSO to remove a specific message, currently configured in a Charging Station.
4	Description	This use case describes how a CSO can remove a specific message, configured via OCPP in a Charging Station.
	Actors	CSO, CSMS, Charging Station
	Scenario description	<ol style="list-style-type: none"> 1. The CSO asks the CSMS to remove a specific message. 2. The CSMS sends a ClearDisplayMessageRequest message with the id of the specific message to the Charging Station. 3. The Charging Station removes the message. 4. The Charging Station response by sending a ClearDisplayMessageResponse message to the CSMS.
5	Prerequisites	There is a message with the given id configured in the Charging Station
6	Postcondition(s)	The message with the given id is removed from the Charging Station

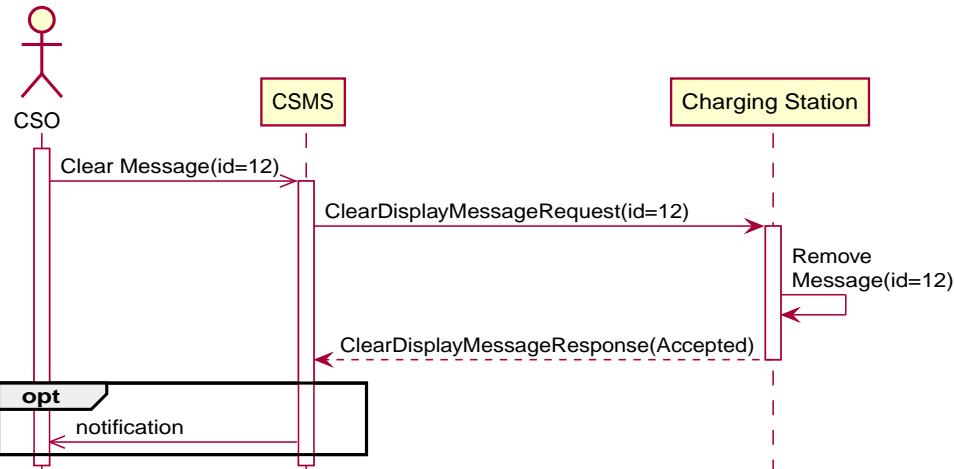


Figure 146. Clear a DisplayMessage sequence diagram

7	Error Handling	n/a
8	Remarks	Only messages configured via OCPP can be cleared/removed via ClearDisplayMessageRequest

005 - Clear a DisplayMessage - Requirements

Table 243. 005 - Clear a DisplayMessage - Requirements

ID	Precondition	Requirement definition
005.FR.01	When a Charging Station receives a ClearDisplayMessageRequest AND there is a message configured in the Charging Station with that id	The Charging Station SHALL respond with a ClearDisplayMessageResponse message with status: Accepted.
005.FR.02	When a Charging Station receives a ClearDisplayMessageRequest AND there is no message configured in the Charging Station with the given id	The Charging Station SHALL respond with a ClearDisplayMessageResponse message with status: Unknown.

006 - Replace DisplayMessage

Table 244. 006 - Replace DisplayMessage

No.	Type	Description
1	Name	Replace DisplayMessage
2	ID	006
	<i>Functional block</i>	O. DisplayMessage
3	Objectives	Enable a CSO to replace DisplayMessages, already configured on a Charging Station.
4	Description	This use case describes how a CSO can replace a DisplayMessage that is previously configured in a Charging Station. Replace the message content, but also all the given parameters with the new one.
	Actors	CSO, CSMS, Charging Station
	Scenario description	<p>1. The CSO asks the CSMS to replace an existing DisplayMessage.</p> <p>2. The CSMS sends a SetDisplayMessageRequest message to the Charging Station with the a DisplayMessage with the same ID as already configured in the Charging Station.</p> <p>3. The Charging Station accepts the request by sending a SetDisplayMessageResponse message to the CSMS.</p> <p>4. The Charging Station shows the updated/replaced message on the display at the configured moment.</p>
	Alternative scenario's	001 - Set DisplayMessage and 002 - Set DisplayMessage for Transaction
5	Prerequisites	There is a message with the same id configured in the Charging Station
6	Postcondition(s)	The DisplayMessage is replaced by the one provided with the same ID.

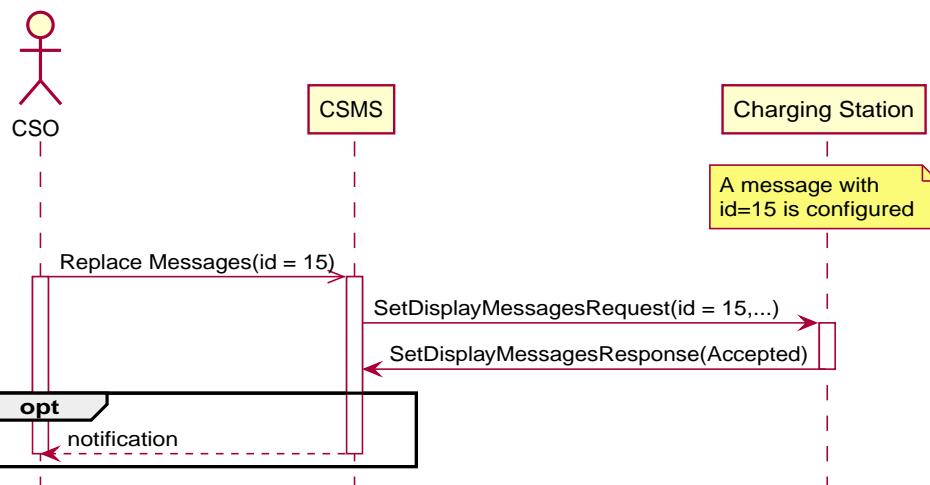


Figure 147. Replace DisplayMessage sequence diagram

7	Error Handling	n/a
8	Remarks	n/a

006 - Replace DisplayMessage - Requirements

Table 245. 006 - Replace DisplayMessage - Requirements

ID	Precondition	Requirement definition
006.FR.01	When a Charging Station receives a SetDisplayMessageRequest AND there is a message configured in the Charging Station with the same id	The Charging Station SHALL replace the existing message with the new message (including all the new parameters) AND respond with a SetDisplayMessageResponse message with status: Accepted for this message.

P. DataTransfer

1. Introduction

This Functional Block describes the functionality that enables parties to extend existing commands with custom attributes or add new custom commands to OCPP.

OCPP offers two mechanisms to create vendor-specific custom extension.

1. The [DataTransferRequest](#) message allows for the exchange of data or messages not standardized in OCPP. As such, it offers a framework within OCPP for experimental functionality that may find its way into future OCPP versions.
Experimenting can be done without creating new (possibly incompatible) OCPP dialects. Secondly, it offers a possibility to implement additional functionality agreed upon between specific CSMS and Charging Station vendors.
2. A CustomData element exists as an optional element in the JSON schemas of all types. CustomData is the only class in the JSON schema files that allows additional properties. It can thus be used to add additional custom attributes to any type.
The CustomData has been deliberately left out of the specification document, because it would introduce a lot of clutter and it is not meant to be used in standard implementations. See also [\[OCPP2.0-PART4\]](#).

The DataTransferRequest/Response contains a field without a length or type specification. It can be convenient to use this field as structured JSON content.

Example of embedded JSON

```
[ 2,  
  "<unique msg id>",  
  "DataTransfer",  
  {  
    "vendorId": "com.mycompany.ice",  
    "messageId": "iceParkedAtCs"  
    "data": { "start_time": "2020-04-01T11:01:02" }  
  }  
]
```

IMPORTANT

Please use with extreme caution and only for optional functionality, since it will impact your compatibility with other systems that do not make use of this option. We recommend mentioning the usage explicitly in your documentation and/or communication. Please consider consulting the Open Charge Alliance before turning to this option to add functionality.

2. Use cases & Requirements

P01 - Data Transfer to the Charging Station

Table 246. P01 - Data Transfer to the Charging Station

No.	Type	Description
1	Name	Data Transfer to the Charging Station
2	ID	P01
	Functional block	P. Data Transfer
3	Objective(s)	To send information from the CSMS to the Charging Station for a function that is not supported by OCPP.
4	Description	This use case covers the functionality of sending a DataTransfer message to the Charging Station from the CSMS.
	Actors	Charging Station, CSMS
	Scenario description	<p>1. The CSMS sends information to a Charging Station for a function not supported by OCPP with DataTransferRequest.</p> <p>2. The Charging Station responds to the CSMS with DataTransferResponse.</p>
5	Prerequisite(s)	n/a
6	Postcondition(s)	<p>Successful postcondition: DataTransferRequest is received <i>Successfully</i> and <i>Accepted</i></p> <p>Failure postcondition: Message has been <i>Accepted</i> but the contained request is <i>Rejected</i>. In all other cases the usage of status <i>Accepted</i> or <i>Rejected</i> and the data element is part of the vendor-specific agreement between the parties involved.</p>

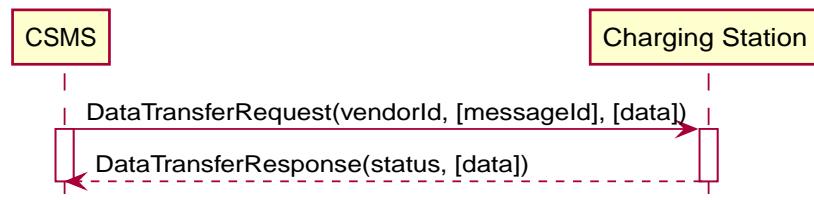


Figure 148. Sequence Diagram: Data Transfer to the Charging Station

7	Error handling	n/a
8	Remark(s)	<p>Data Transfer is used if information for a function is not supported by OCPP.</p> <p>The length of data in both the request and response message is undefined and it is RECOMMENDED that this is agreed upon by all parties involved.</p>

P01 - Data Transfer to the Charging Station - Requirements

Table 247. P01 - Requirements

ID	Precondition	Requirement definition
P01.FR.01		The Charging Station SHALL only use DataTransferRequest for a function which is not supported by OCPP.
P01.FR.02		The vendorId SHOULD be a value from the reversed DNS namespace, where the top tiers of the name, when reversed, should correspond to the publicly registered primary DNS name of the Vendor organization.
P01.FR.03		The messageID in the request message MAY be used to indicate a specific message or implementation.
P01.FR.04		The length of data in both the request and response message is undefined and it is RECOMMENDED that this is agreed upon by all parties involved.

ID	Precondition	Requirement definition
P01.FR.05	If the recipient of the request has no implementation for the specific vendorId.	The recipient SHALL return a status <i>UnknownVendor</i> .
P01.FR.06	Upon receipt of DataTransferRequest and in case of a messageId mismatch (if used).	The recipient SHALL return status <i>UnknownMessageId</i> .
P01.FR.07		The usage of status <i>Accepted</i> or <i>Rejected</i> and the data element SHALL be part of the vendor-specific agreement between the parties involved.

P02 - Data Transfer to the CSMS

Table 248. P02 - Data Transfer to the CSMS

No.	Type	Description
1	Name	Data Transfer to the CSMS
2	ID	P02
	Functional block	P. Data Transfer
3	Objective(s)	To send information from the Charging Station to the CSMS for a function which is not supported by OCPP.
4	Description	This use case covers the functionality of sending a DataTransfer message to the CSMS from the Charging Station.
	Actors	Charging Station, CSMS
	Scenario description	<p>1. The Charging Station sends information to the CSMS for a function not supported by OCPP with DataTransferRequest.</p> <p>2. The CSMS responds to the Charging Station with DataTransferResponse.</p>
5	Prerequisite(s)	n/a
6	Postcondition(s)	<p>Successful postcondition: DataTransferRequest is received Successfully and Accepted</p> <p>Failure postcondition: Message has been accepted but the contained request is <i>Rejected</i>.</p> <p>In all other cases the usage of status <i>Accepted</i> or <i>Rejected</i> and the data element is part of the vendor-specific agreement between the parties involved.</p>

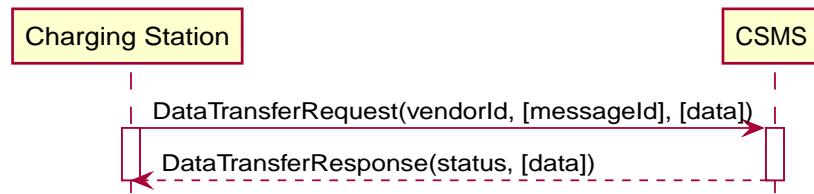


Figure 149. Sequence Diagram: Data Transfer to the CSMS

7	Error handling	n/a
8	Remark(s)	<p>Data Transfer is used if information for a function is <i>not</i> supported by OCPP.</p> <p>The length of data in both the request and response message is undefined and should be agreed upon by all parties involved.</p>

P02 - Data Transfer to the CSMS - Requirements

Table 249. P02 - Requirements

ID	Precondition	Requirement definition
P02.FR.01		The vendorId in the request message SHOULD be known to the Charging Station and uniquely identify the vendor-specific implementation.
P02.FR.02		The Charging Station SHALL only use DataTransferRequest for a function which is not supported by OCPP.
P02.FR.03		The VendorId SHOULD be a value from the reversed DNS namespace, where the top tiers of the name, when reversed, should correspond to the publicly registered primary DNS name of the Vendor organization.
P02.FR.04		The messageId in the request message MAY be used to indicate a specific message or implementation.

ID	Precondition	Requirement definition
P02.FR.05		The length of data in both the request and response message is undefined and it is RECOMMENDED that this is agreed upon by all parties involved.
P02.FR.06	If the recipient of the request has no implementation for the specific vendorId.	The recipient SHALL return a status <i>UnknownVendor</i> .
P02.FR.07	Upon receipt of DataTransferRequest and in case of a messageId mismatch (if used).	The recipient SHALL return status <i>UnknownMessageId</i> .
P02.FR.08		The usage of status Accepted or Rejected and the data element SHALL be part of the vendor-specific agreement between the parties involved.

Messages, Datatypes & Enumerations

1. Messages

1.1. Authorize

1.1.1. AuthorizeRequest

This contains the field definition of the AuthorizeRequest PDU sent by the Charging Station to the CSMS.

Class

Field Name	Field Type	Card.	Description
certificate	string[0..5500]	0..1	Optional. The X.509 certificate chain presented by EV and encoded in PEM format. Order of certificates in chain is from leaf up to (but excluding) root certificate.
idToken	IdTokenType	1..1	Required. This contains the identifier that needs to be authorized.
iso15118CertificateHashData	OCSPRequestDataType	0..4	Optional. Contains the information needed to verify the EV Contract Certificate via OCSP.

1.1.2. AuthorizeResponse

This contains the field definition of the AuthorizeResponse PDU sent by the CSMS to the Charging Station in response to an [AuthorizeRequest](#).

Class

Field Name	Field Type	Card.	Description
certificateStatus	AuthorizeCertificateStatusEnumType	0..1	Optional. Certificate status information. - if all certificates are valid: return 'Accepted'. - if one of the certificates was revoked, return 'CertificateRevoked'.
idTokenInfo	IdTokenInfoType	1..1	Required. This contains information about authorization status, expiry and group id.

1.2. BootNotification

1.2.1. BootNotificationRequest

This contains the field definition of the BootNotificationRequest PDU sent by the Charging Station to the CSMS.

Class

Field Name	Field Type	Card.	Description
reason	BootReasonEnumType	1..1	Required. This contains the reason for sending this message to the CSMS.
chargingStation	ChargingStationType	1..1	Required. Identifies the Charging Station

1.2.2. BootNotificationResponse

This contains the field definition of the BootNotificationResponse PDU sent by the CSMS to the Charging Station in response to a [BootNotificationRequest](#).

Class

Field Name	Field Type	Card.	Description
currentTime	dateTime	1..1	Required. This contains the CSMS's current time.

Field Name	Field Type	Card.	Description
interval	integer	1..1	Required. When Status is Accepted, this contains the heartbeat interval in seconds. If the CSMS returns something other than Accepted, the value of the interval field indicates the minimum wait time before sending a next BootNotification request.
status	RegistrationStatusEnumType	1..1	Required. This contains whether the Charging Station has been registered within the CSMS.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.3. CancelReservation

1.3.1. CancelReservationRequest

This contains the field definition of the CancelReservationRequest PDU sent by the CSMS to the Charging Station.

Class

Field Name	Field Type	Card.	Description
reservationId	integer	1..1	Required. Id of the reservation to cancel.

1.3.2. CancelReservationResponse

This contains the field definition of the CancelReservationResponse PDU sent by the Charging Station to the CSMS in response to a [CancelReservationRequest](#).

Class

Field Name	Field Type	Card.	Description
status	CancelReservationStatusEnumType	1..1	Required. This indicates the success or failure of the canceling of a reservation by CSMS.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.4. CertificateSigned

1.4.1. CertificateSignedRequest

This contains the field definition of the CertificateSignedRequest PDU sent by the CSMS to the Charging Station.

Class

Field Name	Field Type	Card.	Description
certificateChain	string[0..10000]	1..1	Required. The signed PEM encoded X.509 certificate. This SHALL also contain the necessary sub CA certificates, when applicable. The order of the bundle follows the certificate chain, starting from the leaf certificate. The Configuration Variable MaxCertificateChainSize can be used to limit the maximum size of this field.
certificateType	CertificateSigningUseEnumType	0..1	Optional. Indicates the type of the signed certificate that is returned. When omitted the certificate is used for both the 15118 connection (if implemented) and the Charging Station to CSMS connection. This field is required when a certificateType was included in the SignCertificateRequest that requested this certificate to be signed AND both the 15118 connection and the Charging Station connection are implemented.

1.4.2. CertificateSignedResponse

This contains the field definition of the CertificateSignedResponse PDU sent by the Charging Station to the CSMS in response to a [CertificateSignedRequest](#).

Class

Field Name	Field Type	Card.	Description
status	CertificateSignedStatusEnumType	1..1	Required. Returns whether certificate signing has been accepted, otherwise rejected.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.5. ChangeAvailability

1.5.1. ChangeAvailabilityRequest

This contains the field definition of the ChangeAvailabilityRequest PDU sent by the CSMS to the Charging Station.

Class

Field Name	Field Type	Card.	Description
operationalStatus	OperationalStatusEnumType	1..1	Required. This contains the type of availability change that the Charging Station should perform.
evse	EVSEType	0..1	Optional. Contains Id's to designate a specific EVSE/connector by index numbers. When omitted, the message refers to the Charging Station as a whole.

1.5.2. ChangeAvailabilityResponse

This contains the field definition of the ChangeAvailabilityResponse PDU sent by the Charging Station to the CSMS.

Class

Field Name	Field Type	Card.	Description
status	ChangeAvailabilityStatusEnumType	1..1	Required. This indicates whether the Charging Station is able to perform the availability change.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.6. ClearCache

1.6.1. ClearCacheRequest

This contains the field definition of the ClearCacheRequest PDU sent by the CSMS to the Charging Station. No fields are defined.

1.6.2. ClearCacheResponse

This contains the field definition of the ClearCacheResponse PDU sent by the Charging Station to the CSMS in response to a [ClearCacheRequest](#).

Class

Field Name	Field Type	Card.	Description
status	ClearCacheStatusEnumType	1..1	Required. Accepted if the Charging Station has executed the request, otherwise rejected.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.7. ClearChargingProfile

1.7.1. ClearChargingProfileRequest

This contains the field definition of the ClearChargingProfileRequest PDU sent by the CSMS to the Charging Station. The CSMS can use this message to clear (remove) either a specific charging profile (denoted by id) or a selection of charging profiles that match with the values of the optional evse, stackLevel and [ChargingProfilePurpose](#) fields.

Class

Field Name	Field Type	Card.	Description
chargingProfileId	integer	0..1	Optional. The Id of the charging profile to clear.
chargingProfileCriteria	ClearChargingProfileType	0..1	Optional. Specifies the charging profile.

1.7.2. ClearChargingProfileResponse

This contains the field definition of the ClearChargingProfileResponse PDU sent by the Charging Station to the CSMS in response to a ClearChargingProfileRequest.

Class

Field Name	Field Type	Card.	Description
status	ClearChargingProfileStatusEnumType	1..1	Required. Indicates if the Charging Station was able to execute the request.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.8. ClearDisplayMessage

1.8.1. ClearDisplayMessageRequest

This contains the field definition of the ClearDisplayMessageRequest PDU sent by the CSMS to the Charging Station. The CSMS asks the Charging Station to clear a display message that has been configured in the Charging Station to be cleared/removed. See also [005 - Clear a Display Message](#).

Class

Field Name	Field Type	Card.	Description
id	integer	1..1	Required. Id of the message that SHALL be removed from the Charging Station.

1.8.2. ClearDisplayMessageResponse

This contains the field definition of the ClearDisplayMessageResponse PDU sent by the Charging Station to the CSMS in a response to a ClearDisplayMessageRequest. See also [005 - Clear a Display Message](#).

Class

Field Name	Field Type	Card.	Description
status	ClearMessageStatusEnumType	1..1	Required. Returns whether the Charging Station has been able to remove the message.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.9. ClearedChargingLimit

1.9.1. ClearedChargingLimitRequest

This contains the field definition of the ClearedChargingLimitRequest PDU sent by the Charging Station to the CSMS.

Class

Field Name	Field Type	Card.	Description
chargingLimitSource	ChargingLimitSourceEnumType	1..1	Required. Source of the charging limit.
evseld	integer	0..1	Optional. EVSE Identifier.

1.9.2. ClearedChargingLimitResponse

This contains the field definition of the ClearedChargingLimitResponse PDU sent by the CSMS to the Charging Station. No fields are defined.

1.10. ClearVariableMonitoring

1.10.1. ClearVariableMonitoringRequest

This contains the field definition of the ClearVariableMonitoringRequest PDU sent by the CSMS to the Charging Station.

Class

Field Name	Field Type	Card.	Description
id	integer	1..*	Required. List of the monitors to be cleared, identified by their Id.

1.10.2. ClearVariableMonitoringResponse

This contains the field definition of the ClearVariableMonitoringResponse PDU sent by the Charging Station to the CSMS.

Class

Field Name	Field Type	Card.	Description
clearMonitoringResult	ClearMonitoringResultType	1..*	Required. List of result statuses per monitor.

1.11. CostUpdated

1.11.1. CostUpdatedRequest

This contains the field definition of the CostUpdatedRequest PDU sent by the CSMS to the Charging Station. With this request the CSMS can send the current cost of a transaction to a Charging Station.

Class

Field Name	Field Type	Card.	Description
totalCost	decimal	1..1	Required. Current total cost, based on the information known by the CSMS, of the transaction including taxes. In the currency configured with the configuration Variable: [Currency]
transactionId	identifierString[0..36]	1..1	Required. Transaction Id of the transaction the current cost are asked for.

1.11.2. CostUpdatedResponse

This contains the field definition of the CostUpdatedResponse PDU sent by the Charging Station to the CSMS in response to CostUpdatedRequest. No fields are defined.

1.12. CustomerInformation

This contains the field definition of the CustomerInformationRequest PDU sent by the CSMS to the Charging Station.

1.12.1. CustomerInformationRequest

Class

Field Name	Field Type	Card.	Description
requestId	integer	1..1	Required. The Id of the request.
report	boolean	1..1	Required. Flag indicating whether the Charging Station should return NotifyCustomerInformationRequest messages containing information about the customer referred to.
clear	boolean	1..1	Required. Flag indicating whether the Charging Station should clear all information about the customer referred to.
customerIdentifier	string[0..64]	0..1	Optional. A (e.g. vendor specific) identifier of the customer this request refers to. This field contains a custom identifier other than IdToken and Certificate. One of the possible identifiers (customerIdentifier, customerIdToken or customerCertificate) should be in the request message.
idToken	IdTokenType	0..1	Optional. The IdToken of the customer this request refers to. One of the possible identifiers (customerIdentifier, customerIdToken or customerCertificate) should be in the request message.
customerCertificate	CertificateHashDataType	0..1	Optional. The Certificate of the customer this request refers to. One of the possible identifiers (customerIdentifier, customerIdToken or customerCertificate) should be in the request message.

1.12.2. CustomerInformationResponse

Class

Field Name	Field Type	Card.	Description
status	CustomerInformationStatusEnumType	1..1	Required. Indicates whether the request was accepted.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.13. DataTransfer

1.13.1. DataTransferRequest

This contains the field definition of the DataTransferRequest PDU sent either by the CSMS to the Charging Station or vice versa.

Class

Field Name	Field Type	Card.	Description
messageId	string[0..50]	0..1	Optional. May be used to indicate a specific message or implementation.
data	anyType	0..1	Optional. Data without specified length or format. This needs to be decided by both parties (Open to implementation).
vendorId	string[0..255]	1..1	Required. This identifies the Vendor specific implementation

1.13.2. DataTransferResponse

This contains the field definition of the DataTransferResponse PDU sent by the Charging Station to the CSMS or vice versa in response to a [DataTransferRequest](#).

Class

Field Name	Field Type	Card.	Description
status	DataTransferStatusEnumType	1..1	Required. This indicates the success or failure of the data transfer.
data	anyType	0..1	Optional. Data without specified length or format, in response to request.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.14. DeleteCertificate

1.14.1. DeleteCertificateRequest

Used by the CSMS to request deletion of an installed certificate on a Charging Station.

Class

Field Name	Field Type	Card.	Description
certificateHashData	CertificateHashDataType	1..1	Required. Indicates the certificate of which deletion is requested.

1.14.2. DeleteCertificateResponse

Response to a DeleteCertificateRequest.

Class

Field Name	Field Type	Card.	Description
status	DeleteCertificateStatusEnumType	1..1	Required. Charging Station indicates if it can process the request.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.15. FirmwareStatusNotification

1.15.1. FirmwareStatusNotificationRequest

This contains the field definition of the FirmwareStatusNotificationRequest PDU sent by the Charging Station to the CSMS.

Class

Field Name	Field Type	Card.	Description
status	FirmwareStatusEnumType	1..1	Required. This contains the progress status of the firmware installation.
requestId	integer	0..1	Optional. The request id that was provided in the UpdateFirmwareRequest that started this firmware update. This field is mandatory, unless the message was triggered by a TriggerMessageRequest AND there is no firmware update ongoing.

1.15.2. FirmwareStatusNotificationResponse

This contains the field definition of the FirmwareStatusNotificationResponse PDU sent by the CSMS to the Charging Station in response to a [FirmwareStatusNotificationRequest](#). No fields are defined.

1.16. Get15118EVCertificate

1.16.1. Get15118EVCertificateRequest

This message is sent by the Charging Station to the CSMS if an ISO 15118 vehicle selects the service Certificate installation. NOTE: This message is based on CertificateInstallationReq Res from [ISO 15118-2](#).

Class

Field Name	Field Type	Card.	Description
iso15118SchemaVersion	string[0..50]	1..1	Required. Schema version currently used for the 15118 session between EV and Charging Station. Needed for parsing of the EXI stream by the CSMS.
action	CertificateActionEnumType	1..1	Required. Defines whether certificate needs to be installed or updated.
exiRequest	string[0..5600]	1..1	Required. Raw CertificateInstallationReq request from EV, Base64 encoded.

1.16.2. Get15118EVCertificateResponse

Response message from CSMS to Charging Station containing the status and optionally new certificate. NOTE: This message is based on CertificateInstallationReq Res from [ISO 15118-2](#).

Class

Field Name	Field Type	Card.	Description
status	Iso15118EVCertificateStatusEnumType	1..1	Required. Indicates whether the message was processed properly.
exiResponse	string[0..5600]	1..1	Required. Raw CertificateInstallationRes response for the EV, Base64 encoded. The Charging Station can let the CSMS know it supports a higher field size by reporting this using the device model as OCPPCommCtrlr.FieldLength["Get15118EVCertificateResponse.exiResponse"] = <New max length>
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.17. GetBaseReport

1.17.1. GetBaseReportRequest

This contains the field definition of the GetBaseReportRequest PDU sent by the CSMS to the Charging Station.

Class

Field Name	Field Type	Card.	Description
requestId	integer	1..1	Required. The Id of the request.
reportBase	ReportBaseEnumType	1..1	Required. This field specifies the report base.

1.17.2. GetBaseReportResponse

This contains the field definition of the GetBaseReportResponse PDU sent by the Charging Station to the CSMS.

Class

Field Name	Field Type	Card.	Description
status	GenericDeviceModelStatusEnumType	1..1	Required. This indicates whether the Charging Station is able to accept this request.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.18. GetCertificateStatus

1.18.1. GetCertificateStatusRequest

This contains the field definition of the GetCertificateStatusRequest PDU sent by the Charging Station to the CSMS.

Class

Field Name	Field Type	Card.	Description
ocsprequestData	OCSPRequestDataType	1..1	Required. Indicates the certificate of which the status is requested.

1.18.2. GetCertificateStatusResponse

This contains the field definition of the GetCertificateStatusResponse PDU sent by the CSMS to the Charging Station.

Class

Field Name	Field Type	Card.	Description
status	GetCertificateStatusEnumType	1..1	Required. This indicates whether the charging station was able to retrieve the OCSP certificate status.
ocspResult	string[0..5500]	0..1	Optional. OCSPResponse class as defined in IETF RFC 6960 . DER encoded (as defined in IETF RFC 6960), and then base64 encoded. MAY only be omitted when status is not Accepted.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.19. GetChargingProfiles

1.19.1. GetChargingProfilesRequest

The message GetChargingProfilesRequest can be used by the CSMS to request installed charging profiles from the Charging Station. The charging profiles will then be reported by the Charging Station via [ReportChargingProfilesRequest](#) messages.

Class

Field Name	Field Type	Card.	Description
requestId	integer	1..1	Required. Reference identification that is to be used by the Charging Station in the ReportChargingProfilesRequest when provided.
evseld	integer	0..1	Optional. For which EVSE installed charging profiles SHALL be reported. If 0, only charging profiles installed on the Charging Station itself (the grid connection) SHALL be reported. If omitted, all installed charging profiles SHALL be reported. Reported charging profiles SHALL match the criteria in field <i>chargingProfile</i> .
chargingProfile	ChargingProfileCriterionType	1..1	Required. Specifies the charging profile.

1.19.2. GetChargingProfilesResponse

This contains the field definition of the GetChargingProfilesResponse PDU sent by the Charging Station to the CSMS in response to a GetChargingProfilesRequest.

Class

Field Name	Field Type	Card.	Description
status	GetChargingProfileStatusEnumType	1..1	Required. This indicates whether the Charging Station is able to process this request and will send ReportChargingProfilesRequest messages.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.20. GetCompositeSchedule

1.20.1. GetCompositeScheduleRequest

This contains the field definition of the GetCompositeScheduleRequest PDU sent by the CSMS to the Charging Station.

Class

Field Name	Field Type	Card.	Description
duration	integer	1..1	Required. Length of the requested schedule in seconds.
chargingRateUnit	ChargingRateUnitEnumType	0..1	Optional. Can be used to force a power or current profile.
evselid	integer	1..1	Required. The ID of the EVSE for which the schedule is requested. When evselid=0, the Charging Station will calculate the expected consumption for the grid connection.

1.20.2. GetCompositeScheduleResponse

This contains the field definition of the GetCompositeScheduleResponse PDU sent by the Charging Station to the CSMS in response to a [GetCompositeScheduleRequest](#).

Class

Field Name	Field Type	Card.	Description
status	GenericStatusEnumType	1..1	Required. The Charging Station will indicate if it was able to process the request
schedule	CompositeScheduleType	0..1	Optional. This field contains the calculated composite schedule. It may only be omitted when this message contains status Rejected.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.21. GetDisplayMessages

1.21.1. GetDisplayMessagesRequest

Class

Field Name	Field Type	Card.	Description
id	integer	0..*	Optional. If provided the Charging Station shall return Display Messages of the given ids. This field SHALL NOT contain more ids than set in NumberOfDisplayMessages.maxLimit
requestId	integer	1..1	Required. The Id of this request.
priority	MessagePriorityEnumType	0..1	Optional. If provided the Charging Station shall return Display Messages with the given priority only.
state	MessageStateEnumType	0..1	Optional. If provided the Charging Station shall return Display Messages with the given state only.

1.21.2. GetDisplayMessagesResponse

Class

Field Name	Field Type	Card.	Description
status	GetDisplayMessagesStatusEnumType	1..1	Required. Indicates if the Charging Station has Display Messages that match the request criteria in the GetDisplayMessagesRequest
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.22. GetInstalledCertificateIds

1.22.1. GetInstalledCertificateIdsRequest

Used by the CSMS to request an overview of the installed certificates on a Charging Station.

Class

Field Name	Field Type	Card.	Description
certificateType	GetCertificateIdUseEnumType	0..*	Optional. Indicates the type of certificates requested. When omitted, all certificate types are requested.

1.22.2. GetInstalledCertificateIdsResponse

Response to a GetInstalledCertificateIdsRequest.

Class

Field Name	Field Type	Card.	Description
status	GetInstalledCertificateStatusEnumType	1..1	Required. Charging Station indicates if it can process the request.
certificateHashDataChain	CertificateHashDataChainType	0..*	Optional. The Charging Station includes the Certificate information for each available certificate.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.23. GetLocalListVersion

1.23.1. GetLocalListVersionRequest

This contains the field definition of the GetLocalListVersionRequest PDU sent by the CSMS to the Charging Station. No fields are defined.

1.23.2. GetLocalListVersionResponse

This contains the field definition of the GetLocalListVersionResponse PDU sent by the Charging Station to CSMS in response to a [GetLocalListVersionRequest](#).

Class

Field Name	Field Type	Card.	Description
versionNumber	integer	1..1	Required. This contains the current version number of the local authorization list in the Charging Station.

1.24. GetLog

1.24.1. GetLogRequest

This contains the field definition of the GetLogRequest PDU sent by the CSMS to the Charging Station.

Class

Field Name	Field Type	Card.	Description
logType	LogEnumType	1..1	Required. This contains the type of log file that the Charging Station should send.
requestId	integer	1..1	Required. The Id of this request
retries	integer	0..1	Optional. This specifies how many times the Charging Station must retry to upload the log before giving up. If this field is not present, it is left to Charging Station to decide how many times it wants to retry. If the value is 0, it means: no retries.
retryInterval	integer	0..1	Optional. The interval in seconds after which a retry may be attempted. If this field is not present, it is left to Charging Station to decide how long to wait between attempts.
log	LogParametersType	1..1	Required. This field specifies the requested log and the location to which the log should be sent.

1.24.2. GetLogResponse

This contains the field definition of the GetLogResponse PDU sent by the Charging Station to the CSMS in response to a GetLogRequest.

Class

Field Name	Field Type	Card.	Description
status	LogStatusEnumType	1..1	Required. This field indicates whether the Charging Station was able to accept the request.
filename	string[0..255]	0..1	Optional. This contains the name of the log file that will be uploaded. This field is not present when no logging information is available.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.25. GetMonitoringReport

1.25.1. GetMonitoringReportRequest

This contains the field definition of the GetMonitoringReportRequest PDU sent by the CSMS to the Charging Station.

Class

Field Name	Field Type	Card.	Description
requestId	integer	1..1	Required. The Id of the request.
monitoringCriteria	MonitoringCriterionEnumType	0..3	Optional. This field contains criteria for components for which a monitoring report is requested
componentVariable	ComponentVariableType	0..*	Optional. This field specifies the components and variables for which a monitoring report is requested.

1.25.2. GetMonitoringReportResponse

This contains the field definition of the GetMonitoringReportResponse PDU sent by the Charging Station to the CSMS.

Class

Field Name	Field Type	Card.	Description
status	GenericDeviceModelStatusEnumType	1..1	Required. This field indicates whether the Charging Station was able to accept the request.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.26. GetReport

1.26.1. GetReportRequest

This contains the field definition of the GetReportRequest PDU sent by the CSMS to the Charging Station.

Class

Field Name	Field Type	Card.	Description
requestId	integer	1..1	Required. The Id of the request.
componentCriteria	ComponentCriterionEnumType	0..4	Optional. This field contains criteria for components for which a report is requested
componentVariable	ComponentVariableType	0..*	Optional. This field specifies the components and variables for which a report is requested.

1.26.2. GetReportResponse

The response to a GetReportRequest, sent by the Charging Station to the CSMS.

Class

Field Name	Field Type	Card.	Description
status	GenericDeviceModelStatusEnumType	1..1	Required. This field indicates whether the Charging Station was able to accept the request.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.27. GetTransactionStatus

1.27.1. GetTransactionStatusRequest

With this message, the CSMS can ask the Charging Station whether it has transaction-related messages waiting to be delivered to the CSMS. When a transactionId is provided, only messages for a specific transaction are asked for.

Class

Field Name	Field Type	Card.	Description
transactionId	IdentifierString[0..36]	0..1	Optional. The Id of the transaction for which the status is requested.

1.27.2. GetTransactionStatusResponse

The response to a GetTransactionStatusRequest, sent by the Charging Station to the CSMS.

Class

Field Name	Field Type	Card.	Description
ongoingIndicator	boolean	0..1	Optional. Whether the transaction is still ongoing.
messagesInQueue	boolean	1..1	Required. Whether there are still message to be delivered.

1.28. GetVariables

1.28.1. GetVariablesRequest

This contains the field definition of the GetVariablesRequest PDU sent by the CSMS to the Charging Station.

Class

Field Name	Field Type	Card.	Description
getVariableData	GetVariableDataType	1..*	Required. List of requested variables.

1.28.2. GetVariablesResponse

This contains the field definition of the GetVariablesResponse PDU sent by the CSMS to the Charging Station in response to GetVariablesRequest.

Class

Field Name	Field Type	Card.	Description
getVariableResult	GetVariableResultType	1..*	Required. List of requested variables and their values.

1.29. Heartbeat

1.29.1. HeartbeatRequest

This contains the field definition of the HeartbeatRequest PDU sent by the Charging Station to the CSMS. No fields are defined.

1.29.2. HeartbeatResponse

This contains the field definition of the HeartbeatResponse PDU sent by the CSMS to the Charging Station in response to a [HeartbeatRequest](#).

Class

Field Name	Field Type	Card.	Description
currentTime	dateTime	1..1	Required. Contains the current time of the CSMS.

1.30. InstallCertificate

1.30.1. InstallCertificateRequest

Used by the CSMS to request installation of a certificate on a Charging Station. Note: This message is not for installing a TLS client certificate in a charging station. The CertificateSignedRequest mechanism is used for that.

Class

Field Name	Field Type	Card.	Description
certificateType	InstallCertificateUseEnumType	1..1	Required. Indicates the certificate type that is sent.
certificate	string[0..5500]	1..1	Required. A PEM encoded X.509 certificate.

1.30.2. InstallCertificateResponse

The response to a InstallCertificateRequest, sent by the Charging Station to the CSMS.

Class

Field Name	Field Type	Card.	Description
status	InstallCertificateStatusEnumType	1..1	Required. Charging Station indicates if installation was successful.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.31. LogStatusNotification

1.31.1. LogStatusNotificationRequest

This contains the field definition of the LogStatusNotificationRequest PDU sent by the Charging Station to the CSMS.

Class

Field Name	Field Type	Card.	Description
status	UploadLogStatusEnumType	1..1	Required. This contains the status of the log upload.
requestId	integer	0..1	Optional. The request id that was provided in GetLogRequest that started this log upload. This field is mandatory, unless the message was triggered by a TriggerMessageRequest AND there is no log upload ongoing.

1.31.2. LogStatusNotificationResponse

This contains the field definition of the LogStatusNotificationResponse PDU sent by the CSMS to the Charging Station in response to LogStatusNotificationRequest. No fields are defined.

1.32. MeterValues

1.32.1. MeterValuesRequest

This contains the field definition of the MeterValuesRequest PDU sent by the Charging Station to the CSMS. This message might be removed in a future version of OCPP. It will be replaced by Device Management Monitoring events.

Class

Field Name	Field Type	Card.	Description
evseld	integer	1..1	Required. This contains a number (>0) designating an EVSE of the Charging Station. '0' (zero) is used to designate the main power meter.
meterValue	MeterValueType	1..*	Required. The sampled meter values with timestamps.

1.32.2. MeterValuesResponse

This contains the field definition of the MeterValuesResponse PDU sent by the CSMS to the Charging Station in response to a [MeterValuesRequest](#) PDU. This message might be removed in a future version of OCPP. It will be replaced by Device Management Monitoring events.

No fields are defined.

1.33. NotifyChargingLimit

1.33.1. NotifyChargingLimitRequest

The message NotifyChargingLimitRequest can be used to communicate a charging limit, set by an external system on the Charging Station (Not installed by the CSO via [SetChargingProfileRequest](#)), to the CSMS.

Class

Field Name	Field Type	Card.	Description
evselid	integer	0..1	Optional. The charging schedule contained in this notification applies to an EVSE. evselid must be > 0.
chargingLimit	ChargingLimitType	1..1	Required. This contains the source of the charging limit and whether it is grid critical.
chargingSchedule	ChargingScheduleType	0..*	Optional. Contains limits for the available power or current over time, as set by the external source.

1.33.2. NotifyChargingLimitResponse

The NotifyChargingLimitResponse message is sent by the CSMS to the Charging Station in response to a NotifyChargingLimitsRequest. No fields are defined.

1.34. NotifyCustomerInformation

This contains the field definition of the NotifyCustomerInformationRequest PDU sent by the Charging Station to the CSMS.

1.34.1. NotifyCustomerInformationRequest

Class

Field Name	Field Type	Card.	Description
data	string[0..512]	1..1	Required. (Part of) the requested data. No format specified in which the data is returned. Should be human readable.
tbc	boolean	0..1	Optional. "to be continued" indicator. Indicates whether another part of the monitoringData follows in an upcoming notifyMonitoringReportRequest message. Default value when omitted is false.
seqNo	integer	1..1	Required. Sequence number of this message. First message starts at 0.
generatedAt	dateTime	1..1	Required. Timestamp of the moment this message was generated at the Charging Station.
requestId	integer	1..1	Required. The Id of the request.

1.34.2. NotifyCustomerInformationResponse

1.35. NotifyDisplayMessages

1.35.1. NotifyDisplayMessagesRequest

This contains the field definition of the NotifyDisplayMessagesRequest PDU sent by the Charging Station to the CSMS.

Class

Field Name	Field Type	Card.	Description
requestId	integer	1..1	Required. The id of the GetDisplayMessagesRequest that requested this message.
tbc	boolean	0..1	Optional. "to be continued" indicator. Indicates whether another part of the report follows in an upcoming NotifyDisplayMessagesRequest message. Default value when omitted is false.
messageInfo	MessageInfoType	0..*	Optional. The requested display message as configured in the Charging Station.

1.35.2. NotifyDisplayMessagesResponse

The NotifyDisplayMessagesResponse message is sent by the CSMS to the Charging Station in response to a NotifyDisplayMessagesRequest. No fields are defined.

1.36. NotifyEVChargingNeeds

1.36.1. NotifyEVChargingNeedsRequest

The Charging Station uses this message to communicate the charging needs as calculated by the EV to the CSMS.

Class

Field Name	Field Type	Card.	Description
maxScheduleTuples	integer	0..1	Optional. Contains the maximum schedule tuples the car supports per schedule.
evseld	integer	1..1	Required. Defines the EVSE and connector to which the EV is connected. Evseld may not be 0.
chargingNeeds	ChargingNeedsType	1..1	Required. The characteristics of the energy delivery required.

1.36.2. NotifyEVChargingNeedsResponse

Response to a NotifyEVChargingNeedsRequest.

Class

Field Name	Field Type	Card.	Description
status	NotifyEVChargingNeedsStatusEnumType	1..1	Required. Returns whether the CSMS has been able to process the message successfully. It does not imply that the evChargingNeeds can be met with the current charging profile.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.37. NotifyEVChargingSchedule

1.37.1. NotifyEVChargingScheduleRequest

The Charging Station uses this message to communicate the charging schedule as calculated by the EV to the CSMS.

Class

Field Name	Field Type	Card.	Description
timeBase	dateTime	1..1	Required. Periods contained in the charging profile are relative to this point in time.
evseld	integer	1..1	Required. The charging schedule contained in this notification applies to an EVSE. Evseld must be > 0.
chargingSchedule	ChargingScheduleType	1..1	Required. Planned energy consumption of the EV over time. Always relative to timeBase.

1.37.2. NotifyEVChargingScheduleResponse

Response to a [NotifyEVChargingScheduleRequest](#) message.

Class

Field Name	Field Type	Card.	Description
status	GenericStatusEnumType	1..1	Required. Returns whether the CSMS has been able to process the message successfully. It does not imply any approval of the charging schedule.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.38. NotifyEvent

1.38.1. NotifyEventRequest

This contains the field definition of the NotifyEventRequest PDU sent by the Charging Station to the CSMS.

Class

Field Name	Field Type	Card.	Description
generatedAt	dateTime	1..1	Required. Timestamp of the moment this message was generated at the Charging Station.
tbc	boolean	0..1	Optional. "to be continued" indicator. Indicates whether another part of the report follows in an upcoming notifyEventRequest message. Default value when omitted is false.
seqNo	integer	1..1	Required. Sequence number of this message. First message starts at 0.
eventData	EventData	1..*	Required. List of EventData. An EventData element contains only the Component, Variable and VariableMonitoring data that caused the event. The list of EventData will usually contain one eventData element, but the Charging Station may decide to group multiple events in one notification. For example, when multiple events triggered at the same time.

1.38.2. NotifyEventResponse

Response to NotifyEventRequest. No fields are defined.

1.39. NotifyMonitoringReport

1.39.1. NotifyMonitoringReportRequest

This contains the field definition of the NotifyMonitoringRequest PDU sent by the Charging Station to the CSMS.

Class

Field Name	Field Type	Card.	Description
requestId	integer	1..1	Required. The id of the GetMonitoringRequest that requested this report.
tbc	boolean	0..1	Optional. "to be continued" indicator. Indicates whether another part of the monitoringData follows in an upcoming notifyMonitoringReportRequest message. Default value when omitted is false.
seqNo	integer	1..1	Required. Sequence number of this message. First message starts at 0.
generatedAt	dateTime	1..1	Required. Timestamp of the moment this message was generated at the Charging Station.
monitor	MonitoringDataType	0..*	Optional. List of MonitoringData containing monitoring settings.

1.39.2. NotifyMonitoringReportResponse

Response to a NotifyMonitoringRequest message. No fields are defined.

1.40. NotifyReport

1.40.1. NotifyReportRequest

This contains the field definition of the NotifyReportRequest PDU sent by the Charging Station to the CSMS.

Class

Field Name	Field Type	Card.	Description
requestId	integer	1..1	Required. The id of the GetReportRequest or GetBaseReportRequest that requested this report
generatedAt	dateTime	1..1	Required. Timestamp of the moment this message was generated at the Charging Station.
tbc	boolean	0..1	Optional. “to be continued” indicator. Indicates whether another part of the report follows in an upcoming notifyReportRequest message. Default value when omitted is false.
seqNo	integer	1..1	Required. Sequence number of this message. First message starts at 0.
reportData	ReportDataType	0..*	Optional. List of ReportData.

1.40.2. NotifyReportResponse

Response to a NotifyReportRequest message. No fields are defined.

1.41. PublishFirmware

1.41.1. PublishFirmwareRequest

This contains the field definition of the PublishFirmwareRequest PDU sent by the CSMS to the Local Controller.

Class

Field Name	Field Type	Card.	Description
location	string[0..512]	1..1	Required. This contains a string containing a URI pointing to a location from which to retrieve the firmware.
retries	integer	0..1	Optional. This specifies how many times the Charging Station must retry to download the firmware before giving up. If this field is not present, it is left to Charging Station to decide how many times it wants to retry. If the value is 0, it means: no retries.
checksum	identifierString[0..32]	1..1	Required. The MD5 checksum over the entire firmware file as a hexadecimal string of length 32.
requestId	integer	1..1	Required. The Id of the request.
retryInterval	integer	0..1	Optional. The interval in seconds after which a retry may be attempted. If this field is not present, it is left to Charging Station to decide how long to wait between attempts.

1.41.2. PublishFirmwareResponse

This contains the field definition of the PublishFirmwareResponse PDU sent by the Local Controller to the CSMS in response to a PublishFirmwareRequest.

Class

Field Name	Field Type	Card.	Description
status	GenericStatusEnumType	1..1	Required. Indicates whether the request was accepted.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.42. PublishFirmwareStatusNotification

1.42.1. PublishFirmwareStatusNotificationRequest

This contains the field definition of the PublishFirmwareStatusNotificationRequest PDU sent by the Charging Station to the CSMS.

Class

Field Name	Field Type	Card.	Description
status	PublishFirmwareStatusEnumType	1..1	Required. This contains the progress status of the publishfirmware installation.
location	string[0..512]	0..*	Optional. Required if status is Published. Can be multiple URI's, if the Local Controller supports e.g. HTTP, HTTPS, and FTP.
requestId	integer	0..1	Optional. The request id that was provided in the PublishFirmwareRequest which triggered this action.

1.42.2. PublishFirmwareStatusNotificationResponse

This contains the field definition of the PublishFirmwareStatusNotificationResponse PDU sent by the CSMS to the Charging station in response to a PublishFirmwareStatusNotificationRequest.

1.43. ReportChargingProfiles

1.43.1. ReportChargingProfilesRequest

Reports charging profiles installed in the Charging Station, as requested via a [GetChargingProfilesRequest](#) message. The charging profile report can be split over multiple ReportChargingProfilesRequest messages, this can be because charging profiles for different charging sources need to be reported, or because there is just too much data for one message.

Class

Field Name	Field Type	Card.	Description
requestId	integer	1..1	Required. Id used to match the GetChargingProfilesRequest message with the resulting ReportChargingProfilesRequest messages. When the CSMS provided a requestId in the GetChargingProfilesRequest , this field SHALL contain the same value.
chargingLimitSource	ChargingLimitSourceEnumType	1..1	Required. Source that has installed this charging profile.
tbc	boolean	0..1	Optional. To Be Continued. Default value when omitted: false. false indicates that there are no further messages as part of this report.
evseld	integer	1..1	Required. The evse to which the charging profile applies. If evseld = 0, the message contains an overall limit for the Charging Station.
chargingProfile	ChargingProfileType	1..*	Required. The charging profile as configured in the Charging Station.

1.43.2. ReportChargingProfilesResponse

The ReportChargingProfilesResponse message is sent by the CSMS to the Charging Station in response to a [ReportChargingProfilesRequest](#). No fields are defined.

1.44. RequestStartTransaction

1.44.1. RequestStartTransactionRequest

This contains the field definitions of the RequestStartTransactionRequest PDU sent to Charging Station by CSMS.

Class

Field Name	Field Type	Card.	Description
evseld	integer	0..1	Optional. Number of the EVSE on which to start the transaction. Evseld SHALL be > 0
remoteStartId	integer	1..1	Required. Id given by the server to this start request. The Charging Station will return this in the TransactionEventRequest , letting the server know which transaction was started for this request.
idToken	IdTokenType	1..1	Required. The identifier that the Charging Station must use to start a transaction.
chargingProfile	ChargingProfileType	0..1	Optional. Charging Profile to be used by the Charging Station for the requested transaction. ChargingProfilePurpose MUST be set to TxProfile
groupIdToken	IdTokenType	0..1	Optional. The groupIdToken is only relevant when the transaction is to be started on an EVSE for which a reservation for groupIdToken is active, and the configuration variable AuthorizeRemoteStart = false (otherwise the AuthorizeResponse could return the groupIdToken).

1.44.2. RequestStartTransactionResponse

This contains the field definitions of the RequestStartTransactionResponse PDU sent from Charging Station to CSMS.

Class

Field Name	Field Type	Card.	Description
status	RequestStartStopStatusEnumType	1..1	Required. Status indicating whether the Charging Station accepts the request to start a transaction.
transactionId	identifierString[0..36]	0..1	Optional. When the transaction was already started by the Charging Station before the RequestStartTransactionRequest was received, for example: cable plugged in first. This contains the transactionId of the already started transaction.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.45. RequestStopTransaction

1.45.1. RequestStopTransactionRequest

This contains the field definitions of the RequestStopTransactionRequest PDU sent to Charging Station by CSMS.

Class

Field Name	Field Type	Card.	Description
transactionId	identifierString[0..36]	1..1	Required. The identifier of the transaction which the Charging Station is requested to stop.

1.45.2. RequestStopTransactionResponse

This contains the field definitions of the RequestStopTransactionResponse PDU sent from Charging Station to CSMS.

Class

Field Name	Field Type	Card.	Description
status	RequestStartStopStatusEnumType	1..1	Required. Status indicating whether Charging Station accepts the request to stop a transaction.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.46. ReservationStatusUpdate

1.46.1. ReservationStatusUpdateRequest

This contains the field definition of the ReservationStatusUpdateRequest PDU sent by the Charging Station to the CSMS.

Class

Field Name	Field Type	Card.	Description
reservationId	integer	1..1	Required. The ID of the reservation.
reservationUpdateStatus	ReservationUpdateStatusEnumType	1..1	Required. The updated reservation status.

1.46.2. ReservationStatusUpdateResponse

This contains the field definition of the ReservationStatusUpdateResponse PDU sent by the CSMS to the Charging Station in response to a ReservationStatusUpdateRequest. No fields are defined.

1.47. ReserveNow

1.47.1. ReserveNowRequest

This contains the field definition of the ReserveNowRequest PDU sent by the CSMS to the Charging Station.

Class

Field Name	Field Type	Card.	Description
id	integer	1..1	Required. Id of reservation.
expiryDateTime	dateTime	1..1	Required. Date and time at which the reservation expires.
connectorType	ConnectorEnumType	0..1	Optional. This field specifies the connector type.
evseld	integer	0..1	Optional. This contains ID of the evse to be reserved.
idToken	IdTokenType	1..1	Required. The identifier for which the reservation is made.
groupIdToken	IdTokenType	0..1	Optional. The group identifier for which the reservation is made.

1.47.2. ReserveNowResponse

This contains the field definition of the ReserveNowResponse PDU sent by the Charging Station to the CSMS in response to ReserveNowRequest PDU.

Class

Field Name	Field Type	Card.	Description
status	ReserveNowStatusEnumType	1..1	Required. This indicates the success or failure of the reservation.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.48. Reset

1.48.1. ResetRequest

This contains the field definition of the ResetRequest PDU sent by the CSMS to the Charging Station.

Class

Field Name	Field Type	Card.	Description
type	ResetEnumType	1..1	Required. This contains the type of reset that the Charging Station or EVSE should perform.
evselid	integer	0..1	Optional. This contains the ID of a specific EVSE that needs to be reset, instead of the entire Charging Station.

1.48.2. ResetResponse

This contains the field definition of the ResetResponse PDU sent by the Charging Station to the CSMS in response to ResetRequest.

Class

Field Name	Field Type	Card.	Description
status	ResetStatusEnumType	1..1	Required. This indicates whether the Charging Station is able to perform the reset.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.49. SecurityEventNotification

1.49.1. SecurityEventNotificationRequest

Sent by the Charging Station to the CSMS in case of a security event.

Class

Field Name	Field Type	Card.	Description
type	string[0..50]	1..1	Required. Type of the security event. This value should be taken from the Security events list.
timestamp	dateTime	1..1	Required. Date and time at which the event occurred.
techInfo	string[0..255]	0..1	Optional. Additional information about the occurred security event.

1.49.2. SecurityEventNotificationResponse

Sent by the CSMS to the Charging Station to confirm the receipt of a SecurityEventNotificationRequest message. No fields are defined.

1.50. SendLocalList

1.50.1. SendLocalListRequest

This contains the field definition of the SendLocalListRequest PDU sent by the CSMS to the Charging Station. If no (empty) localAuthorizationList is given and the updateType is Full, all IdTokens are removed from the list. Requesting a Differential update without or with empty localAuthorizationList will have no effect on the list. All IdTokens in the localAuthorizationList MUST be unique, no duplicate values are allowed.

Class

Field Name	Field Type	Card.	Description
versionNumber	integer	1..1	Required. In case of a full update this is the version number of the full list. In case of a differential update it is the version number of the list after the update has been applied.
updateType	UpdateEnumType	1..1	Required. This contains the type of update (full or differential) of this request.
localAuthorizationList	AuthorizationData	0..*	Optional. This contains the Local Authorization List entries.

1.50.2. SendLocalListResponse

This contains the field definition of the SendLocalListResponse PDU sent by the Charging Station to the CSMS in response to [SendLocalListRequest](#) PDU.

Class

Field Name	Field Type	Card.	Description
status	SendLocalListStatusEnumType	1..1	Required. This indicates whether the Charging Station has successfully received and applied the update of the Local Authorization List.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.51. SetChargingProfile

1.51.1. SetChargingProfileRequest

This contains the field definition of the SetChargingProfileRequest PDU sent by the CSMS to the Charging Station. The CSMS uses this message to send charging profiles to a Charging Station.

Class

Field Name	Field Type	Card.	Description
evseld	integer	1..1	Required. For TxDefaultProfile an evseld=0 applies the profile to each individual evse. For ChargingStationMaxProfile and ChargingStationExternalConstraints an evseld=0 contains an overall limit for the whole Charging Station.
chargingProfile	ChargingProfileType	1..1	Required. The charging profile to be set at the Charging Station.

1.51.2. SetChargingProfileResponse

This contains the field definition of the SetChargingProfileResponse PDU sent by the Charging Station to the CSMS in response to SetChargingProfileRequest PDU.

Class

Field Name	Field Type	Card.	Description
status	ChargingProfileStatusEnumType	1..1	Required. Returns whether the Charging Station has been able to process the message successfully. This does not guarantee the schedule will be followed to the letter. There might be other constraints the Charging Station may need to take into account.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.52. SetDisplayMessage

1.52.1. SetDisplayMessageRequest

This contains the field definition of the SetDisplayMessageRequest PDU sent by the CSMS to the Charging Station. The CSMS asks the Charging Station to configure a new display message that the Charging Station will display (in the future). See also [001 - Set Display Message](#), [002 - Set Display Message for Transaction](#) and [006 - Replace Display Message](#)

Class

Field Name	Field Type	Card.	Description
message	MessageInfoType	1..1	Required. Message to be configured in the Charging Station, to be displayed.

1.52.2. SetDisplayMessageResponse

This contains the field definition of the SetDisplayMessageResponse PDU sent by the Charging Station to the CSMS in a response to a SetDisplayMessageRequest. See also [001 - Set Display Message](#), [002 - Set Display Message for Transaction](#) and [006 - Replace Display Message](#)

Class

Field Name	Field Type	Card.	Description
status	DisplayMessageStatusEnumType	1..1	Required. This indicates whether the Charging Station is able to display the message.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.53. SetMonitoringBase

1.53.1. SetMonitoringBaseRequest

This contains the field definition of the SetMonitoringBaseRequest PDU sent by the CSMS to the Charging Station.

Class

Field Name	Field Type	Card.	Description
monitoringBase	MonitoringBaseEnumType	1..1	Required. Specify which monitoring base will be set

1.53.2. SetMonitoringBaseResponse

This contains the field definition of the SetMonitoringBaseResponse PDU sent by the Charging Station to the CSMS in response to a SetMonitoringBaseRequest.

Class

Field Name	Field Type	Card.	Description
status	GenericDeviceModelStatusEnumType	1..1	Required. Indicates whether the Charging Station was able to accept the request.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.54. SetMonitoringLevel

1.54.1. SetMonitoringLevelRequest

This contains the field definition of the SetMonitoringLevelRequest PDU sent by the CSMS to the Charging Station.

Class

Field Name	Field Type	Card.	Description
severity	integer	1..1	<p>Required. The Charging Station SHALL only report events with a severity number lower than or equal to this severity. The severity range is 0-9, with 0 as the highest and 9 as the lowest severity level.</p> <p>The severity levels have the following meaning:</p> <p>0-Danger Indicates lives are potentially in danger. Urgent attention is needed and action should be taken immediately.</p> <p>1-Hardware Failure Indicates that the Charging Station is unable to continue regular operations due to Hardware issues. Action is required.</p> <p>2-System Failure Indicates that the Charging Station is unable to continue regular operations due to software or minor hardware issues. Action is required.</p> <p>3-Critical Indicates a critical error. Action is required.</p> <p>4-Error Indicates a non-urgent error. Action is required.</p> <p>5-Alert Indicates an alert event. Default severity for any type of monitoring event.</p> <p>6-Warning Indicates a warning event. Action may be required.</p> <p>7-Notice Indicates an unusual event. No immediate action is required.</p> <p>8-Informational Indicates a regular operational event. May be used for reporting, measuring throughput, etc. No action is required.</p> <p>9-Debug Indicates information useful to developers for debugging, not useful during operations.</p>

1.54.2. SetMonitoringLevelResponse

This contains the field definition of the SetMonitoringLevelResponse PDU sent by the Charging Station to the CSMS in response to a SetMonitoringLevelRequest.

Class

Field Name	Field Type	Card.	Description
status	GenericStatusEnumType	1..1	Required. Indicates whether the Charging Station was able to accept the request.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.55. SetNetworkProfile

1.55.1. SetNetworkProfileRequest

With this message the CSMS gains the ability to configure the connection data (e.g. CSMS URL, OCPP version, APN, etc) on a Charging Station.

Class

Field Name	Field Type	Card.	Description
configurationSlot	integer	1..1	Required. Slot in which the configuration should be stored.
connectionData	NetworkConnectionProfileType	1..1	Required. Connection details.

1.55.2. SetNetworkProfileResponse

This contains the field definition of the SetNetworkProfileResponse PDU sent by the Charging Station to the CSMS in response to a SetNetworkProfileRequest.

Class

Field Name	Field Type	Card.	Description
status	SetNetworkProfileStatusEnumType	1..1	Required. Result of operation.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.56. SetVariableMonitoring

1.56.1. SetVariableMonitoringRequest

This contains the field definition of the SetVariableMonitoringRequest PDU sent by the CSMS to the Charging Station.

Class

Field Name	Field Type	Card.	Description
setMonitoringData	SetMonitoringDataType	1..*	Required. List of MonitoringData containing monitoring settings.

1.56.2. SetVariableMonitoringResponse

This contains the field definition of the SetVariableMonitoringResponse PDU sent by the Charging Station to the CSMS in response to a SetVariableMonitoringRequest.

Class

Field Name	Field Type	Card.	Description
setMonitoringResult	SetMonitoringResultType	1..*	Required. List of result statuses per monitor.

1.57. SetVariables

1.57.1. SetVariablesRequest

This contains the field definition of the SetVariablesRequest PDU sent by the CSMS to the Charging Station.

Class

Field Name	Field Type	Card.	Description
setVariableData	SetVariableDataType	1..*	Required. List of Component-Variable pairs and attribute values to set.

1.57.2. SetVariablesResponse

This contains the field definition of the SetVariablesResponse PDU sent by the Charging Station to the CSMS in response to a SetVariablesRequest.

Class

Field Name	Field Type	Card.	Description
setVariableResult	SetVariableResultType	1..*	Required. List of result statuses per Component-Variable.

1.58. SignCertificate

1.58.1. SignCertificateRequest

Sent by the Charging Station to the CSMS to request that the Certificate Authority signs the public key into a certificate.

Class

Field Name	Field Type	Card.	Description
csr	string[0..5500]	1..1	Required. The Charging Station SHALL send the public key in form of a Certificate Signing Request (CSR) as described in RFC 2986 [22] and then PEM encoded, using the SignCertificateRequest message.
certificateType	CertificateSigningUseEnumType	0..1	Optional. Indicates the type of certificate that is to be signed. When omitted the certificate is to be used for both the 15118 connection (if implemented) and the Charging Station to CSMS connection.

1.58.2. SignCertificateResponse

Sent by the CSMS to the Charging Station in response to the SignCertificateRequest message.

Class

Field Name	Field Type	Card.	Description
status	GenericStatusEnumType	1..1	Required. Specifies whether the CSMS can process the request.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.59. StatusNotification

1.59.1. StatusNotificationRequest

This contains the field definition of the StatusNotificationRequest PDU sent by the Charging Station to the CSMS. This message might be removed in a future version of OCPP. It will be replaced by Device Management Monitoring events.

Class

Field Name	Field Type	Card.	Description
timestamp	dateTime	1..1	Required. The time for which the status is reported.
connectorStatus	ConnectorStatusEnumType	1..1	Required. This contains the current status of the Connector.

Field Name	Field Type	Card.	Description
evseld	integer	1..1	Required. The id of the EVSE to which the connector belongs for which the status is reported.
connectorId	integer	1..1	Required. The id of the connector within the EVSE for which the status is reported.

1.59.2. StatusNotificationResponse

This contains the field definition of StatusNotificationResponse sent by the CSMS to the Charging Station in response to a [StatusNotificationRequest](#). This message might be removed in a future version of OCPP. It will be replaced by Device Management Monitoring events.

No fields are defined.

1.60. TransactionEvent

1.60.1. TransactionEventRequest

This section contains the field definition of the TransactionEventRequest PDU sent by the Charging Station to the CSMS. For each of the eventTypes; Started, Updated and Ended, the corresponding cardinality is specified.

Class

Field Name	Field Type	Card.	Description
eventType	TransactionEventEnumType	1..1	Required. This contains the type of this event. The first TransactionEvent of a transaction SHALL contain: "Started" The last TransactionEvent of a transaction SHALL contain: "Ended" All others SHALL contain: "Updated"
timestamp	dateTime	1..1	Required. The date and time at which this transaction event occurred.
triggerReason	TriggerReasonEnumType	1..1	Required. Reason the Charging Station sends this message to the CSMS
seqNo	integer	1..1	Required. Incremental sequence number, helps with determining if all messages of a transaction have been received.
offline	boolean	0..1	Optional. Indication that this transaction event happened when the Charging Station was offline. Default = false, meaning: the event occurred when the Charging Station was online.
numberOfPhasesUsed	integer	0..1	Optional. If the Charging Station is able to report the number of phases used, then it SHALL provide it. When omitted the CSMS may be able to determine the number of phases used via device management.
cableMaxCurrent	integer	0..1	Optional. The maximum current of the connected cable in Ampere (A).
reservationId	integer	0..1	Optional. This contains the Id of the reservation that terminates as a result of this transaction.
transactionInfo	TransactionType	1..1	Required. Contains transaction specific information.
idToken	IdTokenType	0..1	Optional. This contains the identifier for which a transaction is (or will be) started or stopped. Is required when the EV Driver becomes authorized for this transaction and when the EV Driver ends authorization. The IdToken should only be sent once in a TransactionEventRequest for every authorization (for starting or for stopping) done for this transaction.
evse	EVSEType	0..1	Optional. This identifies which evse (and connector) of the Charging Station is used.

Field Name	Field Type	Card.	Description
meterValue	MeterValueType	0..*	Optional. This contains the relevant meter values. Depending on the EventType of this TransactionEvent the following Configuration Variable is used to configure the content: Started: SampledDataTxStartedMeasurands Updated: SampledDataTxUpdatedMeasurands Ended: SampledDataTxEndedMeasurands & AlignedDataTxEndedMeasurands

1.60.2. TransactionEventResponse

This contains the field definition of the TransactionEventResponse PDU sent by the CSMS to the Charging Station in response to a [TransactionEventRequest](#).

Class

Field Name	Field Type	Card.	Description
totalCost	decimal	0..1	Optional. When eventType of TransactionEventRequest is Updated, then this value contains the <i>running cost</i> . When eventType of TransactionEventRequest is Ended, then this contains the final <i>total cost</i> of this transaction, including taxes, in the currency configured with the Configuration Variable: Currency . Absence of this value does not imply that the transaction was free. To indicate a free transaction, the CSMS SHALL send a value of 0.00.
chargingPriority	integer	0..1	Optional. Priority from a business point of view. Default priority is 0, The range is from -9 to 9. Higher values indicate a higher priority. The chargingPriority in TransactionEventResponse is temporarily, so it may not be set in the IdTokenInfoType afterwards. Also the chargingPriority in TransactionEventResponse overrules the one in IdTokenInfoType .
idTokenInfo	IdTokenInfoType	0..1	Optional. This contains information about authorization status, expiry and group id. Is required when the transactionEventRequest contained an idToken.
updatedPersonalMessage	MessageContentType	0..1	Optional. This can contain updated personal message that can be shown to the EV Driver. This can be used to provide updated tariff information .

1.61. TriggerMessage

1.61.1. TriggerMessageRequest

This contains the field definition of the TriggerMessageRequest PDU sent by the CSMS to the Charging Station.

Class

Field Name	Field Type	Card.	Description
requestedMessage	MessageTriggerEnumType	1..1	Required. Type of message to be triggered.
evse	EVSEType	0..1	Optional. Can be used to specify the EVSE and Connector if required for the message which needs to be sent.

1.61.2. TriggerMessageResponse

This contains the field definition of the TriggerMessageResponse PDU sent by the Charging Station to the CSMS in response to [TriggerMessageResponse](#).

Class

Field Name	Field Type	Card.	Description
status	TriggerMessageStatusEnumType	1..1	Required. Indicates whether the Charging Station will send the requested notification or not.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.62. UnlockConnector

1.62.1. UnlockConnectorRequest

This contains the field definition of the UnlockConnectorRequest PDU sent by the CSMS to the Charging Station.

Class

Field Name	Field Type	Card.	Description
evseld	integer	1..1	Required. This contains the identifier of the EVSE for which a connector needs to be unlocked.
connectorId	integer	1..1	Required. This contains the identifier of the connector that needs to be unlocked.

1.62.2. UnlockConnectorResponse

This contains the field definition of the UnlockConnectorResponse PDU sent by the Charging Station to the CSMS in response to an [UnlockConnectorRequest](#).

Class

Field Name	Field Type	Card.	Description
status	UnlockStatusEnumType	1..1	Required. This indicates whether the Charging Station has unlocked the connector.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.63. UnpublishFirmware

1.63.1. UnpublishFirmwareRequest

This contains the field definition of the UnpublishFirmwareRequest PDU sent by the CSMS to the Charging Station.

Class

Field Name	Field Type	Card.	Description
checksum	identifierString[0..32]	1..1	Required. The MD5 checksum over the entire firmware file as a hexadecimal string of length 32.

1.63.2. UnpublishFirmwareResponse

This contains the field definition of the UnpublishFirmwareResponse PDU sent by the Charging Station to the CSMS in response to a UnpublishFirmwareRequest.

Class

Field Name	Field Type	Card.	Description
status	UnpublishFirmwareStatusEnumType	1..1	Required. Indicates whether the Local Controller succeeded in unpublishing the firmware.

1.64. UpdateFirmware

1.64.1. UpdateFirmwareRequest

This contains the field definition of the UpdateFirmwareRequest PDU sent by the CSMS to the Charging Station.

Class

Field Name	Field Type	Card.	Description
retries	integer	0..1	Optional. This specifies how many times the Charging Station must retry to download the firmware before giving up. If this field is not present, it is left to Charging Station to decide how many times it wants to retry. If the value is 0, it means: no retries.
retryInterval	integer	0..1	Optional. The interval in seconds after which a retry may be attempted. If this field is not present, it is left to Charging Station to decide how long to wait between attempts.
requestId	integer	1..1	Required. The Id of this request
firmware	FirmwareType	1..1	Required. Specifies the firmware to be updated on the Charging Station.

1.64.2. UpdateFirmwareResponse

This contains the field definition of the UpdateFirmwareResponse PDU sent by the Charging Station to the CSMS in response to an [UpdateFirmwareRequest](#).

Class

Field Name	Field Type	Card.	Description
status	UpdateFirmwareStatusEnumType	1..1	Required. This field indicates whether the Charging Station was able to accept the request.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

2. Datatypes

2.1. ACCChargingParametersType

Class

EV AC charging parameters.

ACChargingParametersType is used by: [Common:ChargingNeedsType](#)

Field Name	Field Type	Card.	Description
energyAmount	integer	1..1	Required. Amount of energy requested (in Wh). This includes energy required for preconditioning.
evMinCurrent	integer	1..1	Required. Minimum current (amps) supported by the electric vehicle (per phase).
evMaxCurrent	integer	1..1	Required. Maximum current (amps) supported by the electric vehicle (per phase). Includes cable capacity.
evMaxVoltage	integer	1..1	Required. Maximum voltage supported by the electric vehicle

2.2. AdditionalInfoType

Class

Contains a case insensitive identifier to use for the authorization and the type of authorization to support multiple forms of identifiers.

AdditionalInfoType is used by: [Common:IdTokenType](#)

Field Name	Field Type	Card.	Description
additionalIdToken	identifierString[0..36]	1..1	Required. This field specifies the additional IdToken.
type	string[0..50]	1..1	Required. This defines the type of the additionalIdToken. This is a custom type, so the implementation needs to be agreed upon by all involved parties.

2.3. APNType

Class

Collection of configuration data needed to make a data-connection over a cellular network.

NOTE

When asking a GSM modem to dial in, it is possible to specify which mobile operator should be used. This can be done with the mobile country code (MCC) in combination with a mobile network code (MNC). Example: If your preferred network is Vodafone Netherlands, the MCC=204 and the MNC=04 which means the key PreferredNetwork = 20404 Some modems allows to specify a preferred network, which means, if this network is not available, a different network is used. If you specify UseOnlyPreferredNetwork and this network is not available, the modem will not dial in.

APNType is used by: [SetNetworkProfileRequest.NetworkConnectionProfileType](#)

Field Name	Field Type	Card.	Description
apn	string[0..512]	1..1	Required. The Access Point Name as an URL.
apnUserName	string[0..20]	0..1	Optional. APN username.
apnPassword	string[0..20]	0..1	Optional. APN Password.
simPin	integer	0..1	Optional. SIM card pin code.
preferredNetwork	identifierString[0..6]	0..1	Optional. Preferred network, written as MCC and MNC concatenated. See note.

Field Name	Field Type	Card.	Description
useOnlyPreferredNetwork	boolean	0..1	Optional. Default: false. Use only the preferred Network, do not dial in when not available. See Note.
apnAuthentication	APNAuthenticationEnumType	1..1	Required. Authentication method.

2.4. AuthorizationData

Class

Contains the identifier to use for authorization.

AuthorizationData is used by: [SendLocalListRequest](#)

Field Name	Field Type	Card.	Description
idTokenInfo	IdTokenInfoType	0..1	Optional. Required when UpdateType is Full. This contains information about authorization status, expiry and group id. For a Differential update the following applies: If this element is present, then this entry SHALL be added or updated in the Local Authorization List. If this element is absent, the entry for this IdToken in the Local Authorization List SHALL be deleted.
idToken	IdTokenType	1..1	Required. This contains the identifier which needs to be stored for authorization.

2.5. CertificateHashDataChainType

Class

CertificateHashDataChainType is used by: [GetInstalledCertificateIdsResponse](#)

Field Name	Field Type	Card.	Description
certificateType	GetCertificateIdUseEnumType	1..1	Required. Indicates the type of the requested certificate(s).
certificateHashData	CertificateHashDataType	1..1	Required. Information to identify a certificate.
childCertificateHashData	CertificateHashDataType	0..4	Optional. Information to identify the child certificate(s).

2.6. CertificateHashDataType

Class

CertificateHashDataType is used by: [Common:CertificateHashDataChainType](#) , [DeleteCertificateRequest](#) , [CustomerInformationRequest](#)

Field Name	Field Type	Card.	Description
hashAlgorithm	HashAlgorithmEnumType	1..1	Required. Used algorithms for the hashes provided.
issuerNameHash	identifierString[0..128]	1..1	Required. The hash of the issuer's distinguished name (DN), that must be calculated over the DER encoding of the issuer's name field in the certificate being checked.
issuerKeyHash	string[0..128]	1..1	Required. The hash of the DER encoded public key: the value (excluding tag and length) of the subject public key field in the issuer's certificate.
serialNumber	identifierString[0..40]	1..1	Required. The string representation of the hexadecimal value of the serial number without the prefix "0x" and without leading zeroes.

2.7. ChargingLimitType

Class

ChargingLimitType is used by: [NotifyChargingLimitRequest](#)

Field Name	Field Type	Card.	Description
chargingLimitSource	ChargingLimitSourceEnumType	1..1	Required. Represents the source of the charging limit.
isGridCritical	boolean	0..1	Optional. Indicates whether the charging limit is critical for the grid.

2.8. ChargingNeedsType

Class

ChargingNeedsType is used by: [NotifyEVChargingNeedsRequest](#)

Field Name	Field Type	Card.	Description
requestedEnergyTransfer	EnergyTransferModeEnumType	1..1	Required. Mode of energy transfer requested by the EV.
departureTime	dateTime	0..1	Optional. Estimated departure time of the EV.
acChargingParameters	ACChargingParametersType	0..1	Optional. EV AC charging parameters.
dcChargingParameters	DCCChargingParametersType	0..1	Optional. EV DC charging parameters

2.9. ChargingProfileCriterionType

Class

A ChargingProfile consists of ChargingSchedule, describing the amount of power or current that can be delivered per time interval.

ChargingProfileCriterionType is used by: [GetChargingProfilesRequest](#)

Field Name	Field Type	Card.	Description
chargingProfilePurpose	ChargingProfilePurposeEnumType	0..1	Optional. Defines the purpose of the schedule transferred by this profile
stackLevel	integer	0..1	Optional. Value determining level in hierarchy stack of profiles. Higher values have precedence over lower values. Lowest level is 0.
chargingProfileId	integer	0..*	Optional. List of all the chargingProfileIds requested. Any ChargingProfile that matches one of these profiles will be reported. If omitted, the Charging Station SHALL not filter on chargingProfileId. This field SHALL NOT contain more ids than set in ChargingProfileEntries.maxLimit
chargingLimitSource	ChargingLimitSourceEnumType	0..4	Optional. For which charging limit sources, charging profiles SHALL be reported. If omitted, the Charging Station SHALL not filter on chargingLimitSource.

2.10. ChargingProfileType

Class

A ChargingProfile consists of ChargingSchedule, describing the amount of power or current that can be delivered per time interval.

ChargingProfileType is used by: [RequestStartTransactionRequest](#), [SetChargingProfileRequest](#), [ReportChargingProfilesRequest](#)

Field Name	Field Type	Card.	Description
id	integer	1..1	Required. Id of ChargingProfile.
stackLevel	integer	1..1	Required. Value determining level in hierarchy stack of profiles. Higher values have precedence over lower values. Lowest level is 0.
chargingProfilePurpose	ChargingProfilePurposeEnumType	1..1	Required. Defines the purpose of the schedule transferred by this profile
chargingProfileKind	ChargingProfileKindEnumType	1..1	Required. Indicates the kind of schedule.

Field Name	Field Type	Card.	Description
recurrencyKind	RecurrencyKindEnumType	0..1	Optional. Indicates the start point of a recurrence.
validFrom	dateTime	0..1	Optional. Point in time at which the profile starts to be valid. If absent, the profile is valid as soon as it is received by the Charging Station.
validTo	dateTime	0..1	Optional. Point in time at which the profile stops to be valid. If absent, the profile is valid until it is replaced by another profile.
transactionId	identifierString[0..36]	0..1	Optional. SHALL only be included when ChargingProfilePurpose is set to TxProfile in a SetChargingProfileRequest. The transactionId is used to match the profile to a specific transaction.
chargingSchedule	ChargingScheduleType	1..3	Required. Schedule that contains limits for the available power or current over time. In order to support ISO 15118 schedule negotiation, it supports at most three schedules with associated tariff to choose from.

2.11. ChargingSchedulePeriodType

Class

Charging schedule period structure defines a time period in a charging schedule.

ChargingSchedulePeriodType is used by: [Common:ChargingScheduleType](#) , [Common:CompositeScheduleType](#)

Field Name	Field Type	Card.	Description
startPeriod	integer	1..1	Required. Start of the period, in seconds from the start of schedule. The value of StartPeriod also defines the stop time of the previous period.
limit	decimal	1..1	Required. Charging rate limit during the schedule period, in the applicable chargingRateUnit, for example in Amperes (A) or Watts (W). Accepts at most one digit fraction (e.g. 8.1).
numberPhases	integer	0..1	Optional. The number of phases that can be used for charging. For a DC EVSE this field should be omitted. For an AC EVSE a default value of numberPhases = 3 will be assumed if the field is absent.
phaseToUse	integer	0..1	Optional. Values: 1..3, Used if numberPhases=1 and if the EVSE is capable of switching the phase connected to the EV, i.e. ACPhaseSwitchingSupported is defined and true. It's not allowed unless both conditions above are true. If both conditions are true, and phaseToUse is omitted, the Charging Station / EVSE will make the selection on its own.

2.12. ChargingScheduleType

Class

Charging schedule structure defines a list of charging periods, as used in: GetCompositeSchedule.conf and ChargingProfile.

ChargingScheduleType is used by: [Common:ChargingProfileType](#) , [NotifyChargingLimitRequest](#) , [NotifyEVChargingScheduleRequest](#)

Field Name	Field Type	Card.	Description
id	integer	1..1	Required. Identifies the ChargingSchedule.
startSchedule	dateTime	0..1	Optional. Starting point of an absolute or recurring schedule.

Field Name	Field Type	Card.	Description
duration	integer	0..1	Optional. Duration of the charging schedule in seconds. If the duration is left empty, the last period will continue indefinitely or until end of the transaction if chargingProfilePurpose = TxProfile.
chargingRateUnit	ChargingRateUnitEnumType	1..1	Required. The unit of measure Limit is expressed in.
minChargingRate	decimal	0..1	Optional. Minimum charging rate supported by the EV. The unit of measure is defined by the chargingRateUnit. This parameter is intended to be used by a local smart charging algorithm to optimize the power allocation for in the case a charging process is inefficient at lower charging rates. Accepts at most one digit fraction (e.g. 8.1)
chargingSchedulePeriod	ChargingSchedulePeriodType	1..1024	Required. List of ChargingSchedulePeriod elements defining maximum power or current usage over time. The maximum number of periods, that is supported by the Charging Station, if less than 1024, is set by device model variable SmartChargingCtrlr.PeriodsPerSchedule.
salesTariff	SalesTariffType	0..1	Optional. Sales tariff associated with this charging schedule.

2.13. ChargingStationType

Class

The physical system where an Electrical Vehicle (EV) can be charged.

ChargingStationType is used by: [BootNotificationRequest](#)

Field Name	Field Type	Card.	Description
serialNumber	string[0..25]	0..1	Optional. Vendor-specific device identifier.
model	string[0..20]	1..1	Required. Defines the model of the device.
vendorName	string[0..50]	1..1	Required. Identifies the vendor (not necessarily in a unique manner).
firmwareVersion	string[0..50]	0..1	Optional. This contains the firmware version of the Charging Station.
modem	ModemType	0..1	Optional. Defines the functional parameters of a communication link.

2.14. ClearChargingProfileType

Class

A ChargingProfile consists of a ChargingSchedule, describing the amount of power or current that can be delivered per time interval.

ClearChargingProfileType is used by: [ClearChargingProfileRequest](#)

Field Name	Field Type	Card.	Description
evseld	integer	0..1	Optional. Specifies the id of the EVSE for which to clear charging profiles. An evseld of zero (0) specifies the charging profile for the overall Charging Station. Absence of this parameter means the clearing applies to all charging profiles that match the other criteria in the request.
chargingProfilePurpose	ChargingProfilePurposeEnumType	0..1	Optional. Specifies to purpose of the charging profiles that will be cleared, if they meet the other criteria in the request.
stackLevel	integer	0..1	Optional. Specifies the stackLevel for which charging profiles will be cleared, if they meet the other criteria in the request.

2.15. ClearMonitoringResultType

Class

ClearMonitoringResultType is used by: [ClearVariableMonitoringResponse](#)

Field Name	Field Type	Card.	Description
status	ClearMonitoringStatusEnumType	1..1	Required. Result of the clear request for this monitor, identified by its Id.
id	integer	1..1	Required. Id of the monitor of which a clear was requested.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

2.16. ComponentType

Class

A physical or logical component

ComponentType is used by: [Common:ComponentVariableType](#), [Common:MessageInfoType](#), [GetVariablesRequest.GetVariableDataType](#), [GetVariablesResponse.GetVariableResultType](#), [NotifyMonitoringReportRequest.MonitoringDataType](#), [NotifyReportRequest.ReportDataType](#), [SetVariableMonitoringRequest.SetMonitoringDataType](#), [SetVariableMonitoringResponse.SetMonitoringResultType](#), [SetVariablesRequest.SetVariableDataType](#), [SetVariablesResponse.SetVariableResultType](#), [NotifyEventRequest.EventDataType](#)

Field Name	Field Type	Card.	Description
name	identifierString[0..50]	1..1	Required. Name of the component. Name should be taken from the list of standardized component names whenever possible. Case Insensitive. strongly advised to use Camel Case.
instance	identifierString[0..50]	0..1	Optional. Name of instance in case the component exists as multiple instances. Case Insensitive. strongly advised to use Camel Case.
evse	EVSEType	0..1	Optional. Specifies the EVSE when component is located at EVSE level, also specifies the connector when component is located at Connector level.

2.17. ComponentVariableType

Class

Class to report components, variables and variable attributes and characteristics.

ComponentVariableType is used by: [GetMonitoringReportRequest](#), [GetReportRequest](#)

Field Name	Field Type	Card.	Description
component	ComponentType	1..1	Required. Component for which a report of Variable is requested.
variable	VariableType	0..1	Optional. Variable(s) for which the report is requested.

2.18. CompositeScheduleType

Class

CompositeScheduleType is used by: [GetCompositeScheduleResponse](#)

Field Name	Field Type	Card.	Description
evseId	integer	1..1	Required. The ID of the EVSE for which the schedule is requested. When evseId=0, the Charging Station calculated the expected consumption for the grid connection.
duration	integer	1..1	Required. Duration of the schedule in seconds.
scheduleStart	dateTime	1..1	Required. Date and time at which the schedule becomes active. All time measurements within the schedule are relative to this timestamp.
chargingRateUnit	ChargingRateUnitEnumType	1..1	Required. The unit of measure Limit is expressed in.
chargingSchedulePeriod	ChargingSchedulePeriodType	1..*	Required. List of ChargingSchedulePeriod elements defining maximum power or current usage over time.

2.19. ConsumptionCostType

Class

ConsumptionCostType is used by: [Common:SalesTariffEntryType](#)

Field Name	Field Type	Card.	Description
startValue	decimal	1..1	Required. The lowest level of consumption that defines the starting point of this consumption block. The block interval extends to the start of the next interval.
cost	CostType	1..3	Required. This field contains the cost details.

2.20. CostType

Class

CostType is used by: [Common:ConsumptionCostType](#)

Field Name	Field Type	Card.	Description
costKind	CostKindEnumType	1..1	Required. The kind of cost referred to in the message element amount
amount	integer	1..1	Required. The estimated or actual cost per kWh
amountMultiplier	integer	0..1	Optional. Values: -3..3, The amountMultiplier defines the exponent to base 10 (dec). The final value is determined by: amount * 10 ^ amountMultiplier

2.21. DCChargingParametersType

Class

EV DC charging parameters

DCChargingParametersType is used by: [Common:ChargingNeedsType](#)

Field Name	Field Type	Card.	Description
evMaxCurrent	integer	1..1	Required. Maximum current (amps) supported by the electric vehicle. Includes cable capacity.
evMaxVoltage	integer	1..1	Required. Maximum voltage supported by the electric vehicle
energyAmount	integer	0..1	Optional. Amount of energy requested (in Wh). This includes energy required for preconditioning.
evMaxPower	integer	0..1	Optional. Maximum power (in W) supported by the electric vehicle. Required for DC charging.
stateOfCharge	integer, 0 <= val <= 100	0..1	Optional. Energy available in the battery (in percent of the battery capacity)

Field Name	Field Type	Card.	Description
evEnergyCapacity	integer	0..1	Optional. Capacity of the electric vehicle battery (in Wh)
fullSoC	integer, 0 <= val <= 100	0..1	Optional. Percentage of SoC at which the EV considers the battery fully charged. (possible values: 0 - 100)
bulkSoC	integer, 0 <= val <= 100	0..1	Optional. Percentage of SoC at which the EV considers a fast charging process to end. (possible values: 0 - 100)

2.22. EventDataType

Class

Class to report an event notification for a component-variable.

EventDataType is used by: [NotifyEventRequest](#)

Field Name	Field Type	Card.	Description
eventId	integer	1..1	Required. Identifies the event. This field can be referred to as a cause by other events.
timestamp	dateTime	1..1	Required. Timestamp of the moment the report was generated.
trigger	EventTriggerEnumType	1..1	Required. Type of monitor that triggered this event, e.g. exceeding a threshold value.
cause	integer	0..1	Optional. Refers to the Id of an event that is considered to be the cause for this event.
actualValue	string[0..2500]	1..1	Required. Actual value (<i>attributeType Actual</i>) of the variable. The Configuration Variable ReportingValueSize can be used to limit GetVariableResult.attributeValue, VariableAttribute.value and EventData.actualValue. The max size of these values will always remain equal.
techCode	string[0..50]	0..1	Optional. Technical (error) code as reported by component.
techInfo	string[0..500]	0..1	Optional. Technical detail information as reported by component.
cleared	boolean	0..1	Optional. Cleared is set to true to report the clearing of a monitored situation, i.e. a 'return to normal'.
transactionId	identifierString[0..36]	0..1	Optional. If an event notification is linked to a specific transaction, this field can be used to specify its transactionId.
variableMonitoringId	integer	0..1	Optional. Identifies the VariableMonitoring which triggered the event.
eventNotificationType	EventNotificationEnumType	1..1	Required. Specifies the event notification type of the message.
component	ComponentType	1..1	Required. Component for which event is notified.
variable	VariableType	1..1	Required. Variable for which event is notified.

2.23. EVSEType

Class

Electric Vehicle Supply Equipment

EVSEType is used by: [Common:ComponentType](#) , [TriggerMessageRequest](#) , [ChangeAvailabilityRequest](#) , [TransactionEventRequest](#)

Field Name	Field Type	Card.	Description
id	integer	1..1	Required. EVSE Identifier. This contains a number (> 0) designating an EVSE of the Charging Station.

Field Name	Field Type	Card.	Description
connectordId	integer	0..1	Optional. An id to designate a specific connector (on an EVSE) by connector index number.

2.24. FirmwareType

Class

Represents a copy of the firmware that can be loaded/updated on the Charging Station.

FirmwareType is used by: [UpdateFirmwareRequest](#)

Field Name	Field Type	Card.	Description
location	string[0..512]	1..1	Required. URI defining the origin of the firmware.
retrieveDateTime	dateTime	1..1	Required. Date and time at which the firmware shall be retrieved.
installDateTime	dateTime	0..1	Optional. Date and time at which the firmware shall be installed.
signingCertificate	string[0..5500]	0..1	Optional. Certificate with which the firmware was signed. PEM encoded X.509 certificate.
signature	string[0..800]	0..1	Optional. Base64 encoded firmware signature.

2.25. GetVariableDataType

Class

Class to hold parameters for GetVariables request.

GetVariableDataType is used by: [GetVariablesRequest](#)

Field Name	Field Type	Card.	Description
attributeType	AttributeEnumType	0..1	Optional. Attribute type for which value is requested. When absent, default Actual is assumed.
component	ComponentType	1..1	Required. Component for which the Variable is requested.
variable	VariableType	1..1	Required. Variable for which the attribute value is requested.

2.26. GetVariableResultType

Class

Class to hold results of GetVariables request.

GetVariableResultType is used by: [GetVariablesResponse](#)

Field Name	Field Type	Card.	Description
attributeStatus	GetVariableStatusEnumType	1..1	Required. Result status of getting the variable.
attributeType	AttributeEnumType	0..1	Optional. Attribute type for which value is requested. When absent, default Actual is assumed.
attributeValue	string[0..2500]	0..1	Optional. Value of requested attribute type of component-variable. This field can only be empty when the given status is NOT accepted. The Configuration Variable ReportingValueSize can be used to limit GetVariableResult.attributeValue, VariableAttribute.value and EventData.actualValue. The max size of these values will always remain equal.
component	ComponentType	1..1	Required. Component for which the Variable is requested.

Field Name	Field Type	Card.	Description
variable	VariableType	1..1	Required. Variable for which the attribute value is requested.
attributeStatusInfo	StatusInfoType	0..1	Optional. Detailed attribute status information.

2.27. IdTokenInfoType

Class

Contains status information about an identifier. It is advised to not stop charging for a token that expires during charging, as ExpiryDate is only used for caching purposes. If ExpiryDate is not given, the status has no end date.

IdTokenInfoType is used by: [Common:AuthorizationData](#) , [AuthorizeResponse](#) , [TransactionEventResponse](#)

Field Name	Field Type	Card.	Description
status	AuthorizationStatusEnumType	1..1	Required. Current status of the ID Token.
cacheExpiryDateTime	dateTime	0..1	Optional. Date and Time after which the token must be considered invalid.
chargingPriority	integer	0..1	Optional. Priority from a business point of view. Default priority is 0, The range is from -9 to 9. Higher values indicate a higher priority. The chargingPriority in TransactionEventResponse overrules this one.
language1	string[0..8]	0..1	Optional. Preferred user interface language of identifier user. Contains a language code as defined in [RFC5646] .
evseld	integer	0..*	Optional. Only used when the IdToken is only valid for one or more specific EVSEs, not for the entire Charging Station.
language2	string[0..8]	0..1	Optional. Second preferred user interface language of identifier user. Don't use when language1 is omitted, has to be different from language1. Contains a language code as defined in [RFC5646] .
groupIdToken	IdTokenType	0..1	Optional. This contains the group identifier.
personalMessage	MessageContentType	0..1	Optional. Personal message that can be shown to the EV Driver and can be used for tariff information, user greetings etc.

2.28. IdTokenType

Class

Contains a case insensitive identifier to use for the authorization and the type of authorization to support multiple forms of identifiers.

IdTokenType is used by: [Common:AuthorizationData](#) , [Common:IdTokenInfoType](#) , [RequestStartTransactionRequest](#) , [AuthorizeRequest](#) , [TransactionEventRequest](#) , [ReserveNowRequest](#) , [CustomerInformationRequest](#)

Field Name	Field Type	Card.	Description
idToken	identifierString[0..36]	1..1	Required. IdToken is case insensitive. Might hold the hidden id of an RFID tag, but can for example also contain a UUID.
type	IdTokenEnumType	1..1	Required. Enumeration of possible idToken types.
additionalInfo	AdditionalInfoType	0..*	Optional. AdditionalInfo can be used to send extra information which can be validated by the CSMS in addition to the regular authorization with <i>IdToken</i> . AdditionalInfo contains one or more custom types, which need to be agreed upon by all parties involved. When AdditionalInfo is NOT implemented or a not supported AdditionalInfo.type is used, the CSMS/Charging Station MAY ignore the AdditionalInfo.

2.29. LogParametersType

Class

Generic class for the configuration of logging entries.

LogParametersType is used by: [GetLogRequest](#)

Field Name	Field Type	Card.	Description
remoteLocation	string[0..512]	1..1	Required. The URL of the location at the remote system where the log should be stored.
oldestTimestamp	dateTime	0..1	Optional. This contains the date and time of the oldest logging information to include in the diagnostics.
latestTimestamp	dateTime	0..1	Optional. This contains the date and time of the latest logging information to include in the diagnostics.

2.30. MessageContentType

Class

Contains message details, for a message to be displayed on a Charging Station.

MessageContentType is used by: [Common:IdTokenInfoType](#) , [Common:MessageInfoType](#) , [TransactionEventResponse](#)

Field Name	Field Type	Card.	Description
format	MessageFormatEnumType	1..1	Required. Format of the message.
language	string[0..8]	0..1	Optional. Message language identifier. Contains a language code as defined in [RFC5646] .
content	string[0..512]	1..1	Required. Message contents.

2.31. MessageInfoType

Class

Contains message details, for a message to be displayed on a Charging Station.

MessageInfoType is used by: [SetDisplayMessageRequest](#) , [NotifyDisplayMessagesRequest](#)

Field Name	Field Type	Card.	Description
id	integer	1..1	Required. Unique id within an exchange context. It is defined within the OCPP context as a positive Integer value (greater or equal to zero).
priority	MessagePriorityEnumType	1..1	Required. With what priority should this message be shown
state	MessageStateEnumType	0..1	Optional. During what state should this message be shown. When omitted this message should be shown in any state of the Charging Station.
startDateTime	dateTime	0..1	Optional. From what date-time should this message be shown. If omitted: directly.
endDateTime	dateTime	0..1	Optional. Until what date-time should this message be shown, after this date/time this message SHALL be removed.
transactionId	identifierString[0..36]	0..1	Optional. During which transaction shall this message be shown. Message SHALL be removed by the Charging Station after transaction has ended.
message	MessageContentType	1..1	Required. Contains message details for the message to be displayed on a Charging Station.

Field Name	Field Type	Card.	Description
display	ComponentType	0..1	Optional. When a Charging Station has multiple Displays, this field can be used to define to which Display this message belongs.

2.32. MeterValueType

Class

Collection of one or more sampled values in MeterValuesRequest and TransactionEvent. All sampled values in a MeterValue are sampled at the same point in time.

MeterValueType is used by: [MeterValuesRequest](#), [TransactionEventRequest](#)

Field Name	Field Type	Card.	Description
timestamp	dateTime	1..1	Required. Timestamp for measured value(s).
sampledValue	SampledValueType	1..*	Required. One or more measured values

2.33. ModemType

Class

Defines parameters required for initiating and maintaining wireless communication with other devices.

ModemType is used by: [BootNotificationRequest.ChargingStationType](#)

Field Name	Field Type	Card.	Description
iccid	identifierString[0..20]	0..1	Optional. This contains the ICCID of the modem's SIM card.
imsi	identifierString[0..20]	0..1	Optional. This contains the IMSI of the modem's SIM card.

2.34. MonitoringDataType

Class

Class to hold parameters of SetVariableMonitoring request.

MonitoringDataType is used by: [NotifyMonitoringReportRequest](#)

Field Name	Field Type	Card.	Description
component	ComponentType	1..1	Required. Component for which monitoring report was requested.
variable	VariableType	1..1	Required. Variable for which monitoring report was requested.
variableMonitoring	VariableMonitoringType	1..*	Required. List of monitors for this Component-Variable pair.

2.35. NetworkConnectionProfileType

Class

The NetworkConnectionProfile defines the functional and technical parameters of a communication link.

NetworkConnectionProfileType is used by: [SetNetworkProfileRequest](#)

Field Name	Field Type	Card.	Description
ocppVersion	OCPPVersionEnumType	1..1	Required. Defines the OCPP version used for this communication function.
ocppTransport	OCPPTransportEnumType	1..1	Required. Defines the transport protocol (e.g. SOAP or JSON). Note: SOAP is not supported in OCPP 2.0, but is supported by other versions of OCPP.
ocppCsmsUrl	string[0..512]	1..1	Required. URL of the CSMS(s) that this Charging Station communicates with.
messageTimeout	integer	1..1	Required. Duration in seconds before a message send by the Charging Station via this network connection times-out. The best setting depends on the underlying network and response times of the CSMS. If you are looking for a some guideline: use 30 seconds as a starting point.
securityProfile	integer	1..1	Required. This field specifies the security profile used when connecting to the CSMS with this NetworkConnectionProfile.
ocppInterface	OCPPInterfaceEnumType	1..1	Required. Applicable Network Interface.
vpn	VPNType	0..1	Optional. Settings to be used to set up the VPN connection
apn	APNType	0..1	Optional. Collection of configuration data needed to make a data-connection over a cellular network.

2.36. OCSPRequestDataType

Class

OCSPRequestDataType is used by: [AuthorizeRequest](#), [GetCertificateStatusRequest](#)

Field Name	Field Type	Card.	Description
hashAlgorithm	HashAlgorithmEnumType	1..1	Required. Used algorithms for the hashes provided.
issuerNameHash	identifierString[0..128]	1..1	Required. The hash of the issuer's distinguished name (DN), that must be calculated over the DER encoding of the issuer's name field in the certificate being checked.
issuerKeyHash	string[0..128]	1..1	Required. The hash of the DER encoded public key: the value (excluding tag and length) of the subject public key field in the issuer's certificate.
serialNumber	identifierString[0..40]	1..1	Required. The string representation of the hexadecimal value of the serial number without the prefix "0x" and without leading zeroes.
responderURL	string[0..512]	1..1	Required. This contains the responder URL (Case insensitive).

2.37. RelativeTimeIntervalType

Class

RelativeTimeIntervalType is used by: [Common:SalesTariffEntryType](#)

Field Name	Field Type	Card.	Description
start	integer	1..1	Required. Start of the interval, in seconds from NOW.
duration	integer	0..1	Optional. Duration of the interval, in seconds.

2.38. ReportDataType

Class

Class to report components, variables and variable attributes and characteristics.

ReportDataType is used by: [NotifyReportRequest](#)

Field Name	Field Type	Card.	Description
component	ComponentType	1..1	Required. Component for which a report of Variable is requested.
variable	VariableType	1..1	Required. Variable for which report is requested.
variableAttribute	VariableAttributeType	1..4	Required. Attribute data of a variable.
variableCharacteristics	VariableCharacteristicsType	0..1	Optional. Fixed read-only parameters of a variable.

2.39. SalesTariffEntryType

Class

SalesTariffEntryType is used by: [Common:SalesTariffType](#)

Field Name	Field Type	Card.	Description
ePriceLevel	integer, 0 <= val	0..1	Optional. Defines the price level of this SalesTariffEntry (referring to NumEPriceLevels). Small values for the EPriceLevel represent a cheaper TariffEntry. Large values for the EPriceLevel represent a more expensive TariffEntry.
relativeTimeInterval	RelativeTimeIntervalType	1..1	Required. Defines the time interval the SalesTariffEntry is valid for, based upon relative times.
consumptionCost	ConsumptionCostType	0..3	Optional. Defines additional means for further relative price information and/or alternative costs.

2.40. SalesTariffType

Class

NOTE This dataType is based on dataTypes from [ISO 15118-2](#).

SalesTariffType is used by: [Common:ChargingScheduleType](#)

Field Name	Field Type	Card.	Description
id	integer	1..1	Required. SalesTariff identifier used to identify one sales tariff. An SAID remains a unique identifier for one schedule throughout a charging session.
salesTariffDescription	string[0..32]	0..1	Optional. A human readable title/short description of the sales tariff e.g. for HMI display purposes.
numEPriceLevels	integer	0..1	Optional. Defines the overall number of distinct price levels used across all provided SalesTariff elements.
salesTariffEntry	SalesTariffEntryType	1..1024	Required. Encapsulating element describing all relevant details for one time interval of the SalesTariff. The number of SalesTariffEntry elements is limited by the parameter maxScheduleTuples.

2.41. SampledValueType

Class

Single sampled value in MeterValues. Each value can be accompanied by optional fields.

To save on mobile data usage, default values of all of the optional fields are such that. The value without any additional fields will be interpreted, as a register reading of active import energy in Wh (Watt-hour) units.

SampledValueType is used by: [Common:MeterValueType](#)

Field Name	Field Type	Card.	Description
value	decimal	1..1	Required. Indicates the measured value.

Field Name	Field Type	Card.	Description
context	ReadingContextEnumType	0..1	Optional. Type of detail value: start, end or sample. Default = "Sample.Periodic"
measurand	MeasurandEnumType	0..1	Optional. Type of measurement. Default = "Energy.Active.Import.Register"
phase	PhaseEnumType	0..1	Optional. Indicates how the measured value is to be interpreted. For instance between L1 and neutral (L1-N) Please note that not all values of phase are applicable to all Measurands. When phase is absent, the measured value is interpreted as an overall value.
location	LocationEnumType	0..1	Optional. Indicates where the measured value has been sampled. Default = "Outlet"
signedMeterValue	SignedMeterValueType	0..1	Optional. Contains the MeterValueSignature with sign/encoding method information.
unitOfMeasure	UnitOfMeasureType	0..1	Optional. Represents a UnitOfMeasure including a multiplier

2.42. SetMonitoringDataType

Class

Class to hold parameters of SetVariableMonitoring request.

SetMonitoringDataType is used by: [SetVariableMonitoringRequest](#)

Field Name	Field Type	Card.	Description
id	integer	0..1	Optional. An id SHALL only be given to replace an existing monitor. The Charging Station handles the generation of id's for new monitors.
transaction	boolean	0..1	Optional. Monitor only active when a transaction is ongoing on a component relevant to this transaction. Default = false.
value	decimal	1..1	Required. Value for threshold or delta monitoring. For Periodic or PeriodicClockAligned this is the interval in seconds.
type	MonitorEnumType	1..1	Required. The type of this monitor, e.g. a threshold, delta or periodic monitor.

Field Name	Field Type	Card.	Description
severity	integer	1..1	<p>Required. The severity that will be assigned to an event that is triggered by this monitor. The severity range is 0-9, with 0 as the highest and 9 as the lowest severity level.</p> <p>The severity levels have the following meaning:</p> <ul style="list-style-type: none"> 0-Danger Indicates lives are potentially in danger. Urgent attention is needed and action should be taken immediately. 1-Hardware Failure Indicates that the Charging Station is unable to continue regular operations due to Hardware issues. Action is required. 2-System Failure Indicates that the Charging Station is unable to continue regular operations due to software or minor hardware issues. Action is required. 3-Critical Indicates a critical error. Action is required. 4-Error Indicates a non-urgent error. Action is required. 5-Alert Indicates an alert event. Default severity for any type of monitoring event. 6-Warning Indicates a warning event. Action may be required. 7-Notice Indicates an unusual event. No immediate action is required. 8-Informational Indicates a regular operational event. May be used for reporting, measuring throughput, etc. No action is required. 9-Debug Indicates information useful to developers for debugging, not useful during operations.
component	ComponentType	1..1	Required. Component for which monitor is set.
variable	VariableType	1..1	Required. Variable for which monitor is set.

2.43. SetMonitoringResultType

Class

Class to hold result of SetVariableMonitoring request.

SetMonitoringResultType is used by: [SetVariableMonitoringResponse](#)

Field Name	Field Type	Card.	Description
id	integer	0..1	Optional. Id given to the VariableMonitor by the Charging Station. The Id is only returned when status is accepted. Installed VariableMonitors should have unique id's but the id's of removed Installed monitors should have unique id's but the id's of removed monitors MAY be reused.
status	SetMonitoringStatusEnumType	1..1	Required. Status is OK if a value could be returned. Otherwise this will indicate the reason why a value could not be returned.
type	MonitorEnumType	1..1	Required. The type of this monitor, e.g. a threshold, delta or periodic monitor.

Field Name	Field Type	Card.	Description
severity	integer	1..1	<p>Required. The severity that will be assigned to an event that is triggered by this monitor. The severity range is 0-9, with 0 as the highest and 9 as the lowest severity level.</p> <p>The severity levels have the following meaning:</p> <ul style="list-style-type: none"> 0-Danger Indicates lives are potentially in danger. Urgent attention is needed and action should be taken immediately. 1-Hardware Failure Indicates that the Charging Station is unable to continue regular operations due to Hardware issues. Action is required. 2-System Failure Indicates that the Charging Station is unable to continue regular operations due to software or minor hardware issues. Action is required. 3-Critical Indicates a critical error. Action is required. 4-Error Indicates a non-urgent error. Action is required. 5-Alert Indicates an alert event. Default severity for any type of monitoring event. 6-Warning Indicates a warning event. Action may be required. 7-Notice Indicates an unusual event. No immediate action is required. 8-Informational Indicates a regular operational event. May be used for reporting, measuring throughput, etc. No action is required. 9-Debug Indicates information useful to developers for debugging, not useful during operations.
component	ComponentType	1..1	Required. Component for which status is returned.
variable	VariableType	1..1	Required. Variable for which status is returned.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

2.44. SetVariableDataType

Class

SetVariableDataType is used by: [SetVariablesRequest](#)

Field Name	Field Type	Card.	Description
attributeType	AttributeEnumType	0..1	Optional. Type of attribute: Actual, Target, MinSet, MaxSet. Default is Actual when omitted.
attributeValue	string[0..1000]	1..1	Required. Value to be assigned to attribute of variable. The value is allowed to be an empty string (""). The Configuration Variable ConfigurationValueSize can be used to limit SetVariableData.attributeValue and VariableCharacteristics.valueList. The max size of these values will always remain equal.
component	ComponentType	1..1	Required. The component for which the variable data is set.
variable	VariableType	1..1	Required. Specifies the that needs to be set.

2.45. SetVariableResultType

Class

SetVariableResultType is used by: [SetVariablesResponse](#)

Field Name	Field Type	Card.	Description
attributeType	AttributeEnumType	0..1	Optional. Type of attribute: Actual, Target, MinSet, MaxSet. Default is Actual when omitted.
attributeStatus	SetVariableStatusEnumType	1..1	Required. Result status of setting the variable.
component	ComponentType	1..1	Required. The component for which result is returned.
variable	VariableType	1..1	Required. The variable for which the result is returned.
attributeStatusInfo	StatusInfoType	0..1	Optional. Detailed attribute status information.

2.46. SignedMeterValueType

Class

Represent a signed version of the meter value.

SignedMeterValueType is used by: [Common:SampledValueType](#)

Field Name	Field Type	Card.	Description
signedMeterData	string[0..2500]	1..1	Required. Base64 encoded, contains the signed data which might contain more than just the meter value. It can contain information like timestamps, reference to a customer etc.
signingMethod	string[0..50]	1..1	Required. Method used to create the digital signature.
encodingMethod	string[0..50]	1..1	Required. Method used to encode the meter values before applying the digital signature algorithm.
publicKey	string[0..2500]	1..1	Required. Base64 encoded, sending depends on configuration variable <i>PublicKeyWithSignedMeterValue</i> .

2.47. StatusInfoType

Class

Element providing more information about the status.

StatusInfoType is used by: [Common:ClearMonitoringResultType](#) , [BootNotificationResponse](#) , [CancelReservationResponse](#) , [TriggerMessageResponse](#) , [UnlockConnectorResponse](#) , [UpdateFirmwareResponse](#) , [ClearDisplayMessageResponse](#) , [Get15118EVCertificateResponse](#) , [GetCompositeScheduleResponse](#) , [ChangeAvailabilityResponse](#) , [GetLogResponse](#) , [ClearChargingProfileResponse](#) , [NotifyEVChargingNeedsResponse](#) , [ClearCacheResponse](#) , [NotifyEVChargingScheduleResponse](#) , [RequestStartTransactionResponse](#) , [RequestStopTransactionResponse](#) , [SetChargingProfileResponse](#) , [SetDisplayMessageResponse](#) , [SetNetworkProfileResponse](#) , [SignCertificateResponse](#) , [DataTransferResponse](#) , [CertificateSignedResponse](#) , [DeleteCertificateResponse](#) , [GetChargingProfilesResponse](#) , [GetInstalledCertificateIdsResponse](#) , [InstallCertificateResponse](#) , [GetBaseReportResponse](#) , [GetMonitoringReportResponse](#) , [GetReportResponse](#) , [GetVariablesResponse](#).[GetVariableResultType](#) , [ReserveNowResponse](#) , [SetMonitoringBaseResponse](#) , [SetMonitoringLevelResponse](#) , [SetVariableMonitoringResponse](#).[SetMonitoringResultType](#) , [SetVariablesResponse](#).[SetVariableResultType](#) , [PublishFirmwareResponse](#) , [GetCertificateStatusResponse](#) , [ResetResponse](#) , [GetDisplayMessagesResponse](#) , [CustomerInformationResponse](#) , [SendLocalListResponse](#)

Field Name	Field Type	Card.	Description
reasonCode	string[0..20]	1..1	Required. A predefined code for the reason why the status is returned in this response. The string is case-insensitive.
additionalInfo	string[0..512]	0..1	Optional. Additional text to provide detailed information.

2.48. TransactionType

Class

TransactionType is used by: [TransactionEventRequest](#)

Field Name	Field Type	Card.	Description
transactionId	identifierString[0..36]	1..1	Required. This contains the Id of the transaction.
chargingState	ChargingStateEnumType	0..1	Optional. Current charging state, is required when state has changed.
timeSpentCharging	integer	0..1	Optional. Contains the total time that energy flowed from EVSE to EV during the transaction (in seconds). Note that timeSpentCharging is smaller or equal to the duration of the transaction.
stoppedReason	ReasonEnumType	0..1	Optional. This contains the reason why the transaction was stopped. MAY only be omitted when Reason is "Local".
remoteStartId	integer	0..1	Optional. The ID given to remote start request (RequestStartTransactionRequest). This enables to CSMS to match the started transaction to the given start request.

2.49. UnitOfMeasureType

Class

Represents a UnitOfMeasure with a multiplier

UnitOfMeasureType is used by: [Common:SampledValueType](#)

Field Name	Field Type	Card.	Description
unit	string[0..20]	0..1	Optional. Unit of the value. Default = "Wh" if the (default) measurand is an "Energy" type. This field SHALL use a value from the list Standardized Units of Measurements in Part 2 Appendices. If an applicable unit is available in that list, otherwise a "custom" unit might be used.
multiplier	integer	0..1	Optional. Multiplier, this value represents the exponent to base 10. I.e. multiplier 3 means 10 raised to the 3rd power. Default is 0.

2.50. VariableAttributeType

Class

Attribute data of a variable.

VariableAttributeType is used by: [NotifyReportRequest.ReportDataType](#)

Field Name	Field Type	Card.	Description
type	AttributeEnumType	0..1	Optional. Attribute: Actual, MinSet, MaxSet, etc. Defaults to Actual if absent.
value	string[0..2500]	0..1	Optional. Value of the attribute. May only be omitted when mutability is set to 'WriteOnly'. The Configuration Variable ReportingValueSize can be used to limit GetVariableResult.attributeValue, VariableAttribute.value and EventData.actualValue. The max size of these values will always remain equal.
mutability	MutabilityEnumType	0..1	Optional. Defines the mutability of this attribute. Default is ReadWrite when omitted.

Field Name	Field Type	Card.	Description
persistent	boolean	0..1	Optional. If true, value will be persistent across system reboots or power down. Default when omitted is false.
constant	boolean	0..1	Optional. If true, value that will never be changed by the Charging Station at runtime. Default when omitted is false.

2.51. VariableCharacteristicsType

Class

Fixed read-only parameters of a variable.

VariableCharacteristicsType is used by: [NotifyReportRequest.ReportDataType](#)

Field Name	Field Type	Card.	Description
unit	string[0..16]	0..1	Optional. Unit of the variable. When the transmitted value has a unit, this field SHALL be included.
dataType	DataEnumType	1..1	Required. Data type of this variable.
minLimit	decimal	0..1	Optional. Minimum possible value of this variable.
maxLimit	decimal	0..1	Optional. Maximum possible value of this variable. When the datatype of this Variable is String, OptionList, SequenceList or MemberList, this field defines the maximum length of the (CSV) string.
valuesList	string[0..1000]	0..1	Optional. Allowed values when variable is Option/Member/SequenceList. * OptionList: The (Actual) Variable value must be a single value from the reported (CSV) enumeration list. * MemberList: The (Actual) Variable value may be an (unordered) (sub-)set of the reported (CSV) valid values list. * SequenceList: The (Actual) Variable value may be an ordered (priority, etc) (sub-)set of the reported (CSV) valid values list. This is a comma separated list. The Configuration Variable ConfigurationValueSize can be used to limit SetVariableData.attributeValue and VariableCharacteristics.valueList. The max size of these values will always remain equal.
supportsMonitoring	boolean	1..1	Required. Flag indicating if this variable supports monitoring.

2.52. VariableMonitoringType

Class

A monitoring setting for a variable.

VariableMonitoringType is used by: [NotifyMonitoringReportRequest.MonitoringDataType](#)

Field Name	Field Type	Card.	Description
id	integer	1..1	Required. Identifies the monitor.
transaction	boolean	1..1	Required. Monitor only active when a transaction is ongoing on a component relevant to this transaction.

Field Name	Field Type	Card.	Description
value	decimal	1..1	Required. Value for threshold or delta monitoring. For Periodic or PeriodicClockAligned this is the interval in seconds.
type	MonitorEnumType	1..1	Required. The type of this monitor, e.g. a threshold, delta or periodic monitor.
severity	integer	1..1	<p>Required. The severity that will be assigned to an event that is triggered by this monitor. The severity range is 0-9, with 0 as the highest and 9 as the lowest severity level.</p> <p>The severity levels have the following meaning:</p> <p>0-Danger Indicates lives are potentially in danger. Urgent attention is needed and action should be taken immediately.</p> <p>1-Hardware Failure Indicates that the Charging Station is unable to continue regular operations due to Hardware issues. Action is required.</p> <p>2-System Failure Indicates that the Charging Station is unable to continue regular operations due to software or minor hardware issues. Action is required.</p> <p>3-Critical Indicates a critical error. Action is required.</p> <p>4-Error Indicates a non-urgent error. Action is required.</p> <p>5-Alert Indicates an alert event. Default severity for any type of monitoring event.</p> <p>6-Warning Indicates a warning event. Action may be required.</p> <p>7-Notice Indicates an unusual event. No immediate action is required.</p> <p>8-Informational Indicates a regular operational event. May be used for reporting, measuring throughput, etc. No action is required.</p> <p>9-Debug Indicates information useful to developers for debugging, not useful during operations.</p>

2.53. VariableType

Class

Reference key to a component-variable.

VariableType is used by: [Common:ComponentVariableType](#) , [GetVariablesRequest.GetVariableDataType](#) , [GetVariablesResponse.GetVariableResultType](#) , [NotifyMonitoringReportRequest.MonitoringDataType](#) , [NotifyReportRequest.ReportDataType](#) , [SetVariableMonitoringRequest.SetMonitoringDataType](#) , [SetVariableMonitoringResponse.SetMonitoringResultType](#) , [SetVariablesRequest.SetVariableDataType](#) , [SetVariablesResponse.SetVariableResultType](#) , [NotifyEventRequest.EventDataType](#)

Field Name	Field Type	Card.	Description
name	identifierString[0..50]	1..1	Required. Name of the variable. Name should be taken from the list of standardized variable names whenever possible. Case Insensitive. strongly advised to use Camel Case.

Field Name	Field Type	Card.	Description
instance	identifierString[0..50]	0..1	Optional. Name of instance in case the variable exists as multiple instances. Case Insensitive. strongly advised to use Camel Case.

2.54. VPNTYPE

Class

VPN Configuration settings

VPNTYPE is used by: [SetNetworkProfileRequest.NetworkConnectionProfileType](#)

Field Name	Field Type	Card.	Description
server	string[0..512]	1..1	Required. VPN Server Address
user	string[0..20]	1..1	Required. VPN User
group	string[0..20]	0..1	Optional. VPN group.
password	string[0..20]	1..1	Required. VPN Password.
key	string[0..255]	1..1	Required. VPN shared secret.
type	VPNEnumType	1..1	Required. Type of VPN

3. Enumerations

3.1. APNAuthenticationEnumType

Enumeration

APNAuthenticationEnumType is used by: [setNetworkProfile:SetNetworkProfileRequest.APNTyp](#)e

Value	Description
CHAP	Use CHAP authentication
NONE	Use no authentication
PAP	Use PAP authentication
AUTO	Sequentially try CHAP, PAP, NONE.

3.2. AttributeEnumType

Enumeration

AttributeEnumType is used by: [Common:VariableAttributeType](#) , [getVariables:GetVariablesRequest.GetVariableDataType](#) , [getVariables:GetVariablesResponse.GetVariableResultType](#) , [setVariables:SetVariablesRequest.SetVariableDataType](#) , [setVariables:SetVariablesResponse.SetVariableResultType](#)

Value	Description
Actual	The actual value of the variable.
Target	The target value for this variable.
MinSet	The minimal allowed value for this variable
MaxSet	The maximum allowed value for this variable

3.3. AuthorizationStatusEnumType

Enumeration

Status of an authorization response.

AuthorizationStatusEnumType is used by: [Common:IdTokenInfoType](#)

Value	Description
Accepted	Identifier is allowed for charging.
Blocked	Identifier has been blocked. Not allowed for charging.
ConcurrentTx	Identifier is already involved in another transaction and multiple transactions are not allowed. (Only relevant for the response to a transactionEventRequest(eventType=Started).)
Expired	Identifier has expired. Not allowed for charging.
Invalid	Identifier is invalid. Not allowed for charging.
NoCredit	Identifier is valid, but EV Driver doesn't have enough credit to start charging. Not allowed for charging.
NotAllowedTypeEVSE	Identifier is valid, but not allowed to charge at this type of EVSE.
NotAtThisLocation	Identifier is valid, but not allowed to charge at this location.
NotAtThisTime	Identifier is valid, but not allowed to charge at this location at this time.
Unknown	Identifier is unknown. Not allowed for charging.

3.4. AuthorizeCertificateStatusEnumType

Enumeration

Status of the EV Contract certificate.

AuthorizeCertificateStatusEnumType is used by: [authorize:AuthorizeResponse](#)

Value	Description
Accepted	Positive response
SignatureError	If the validation of the Security element in the message header failed.
CertificateExpired	If the OEMProvisioningCert in the CertificateInstallationReq, the Contract Certificate in the CertificateUpdateReq, or the ContractCertificate in the PaymentDetailsReq is expired.
CertificateRevoked	Used when the SECC or CSMS matches the ContractCertificate contained in a CertificateUpdateReq or PaymentDetailsReq with a CRL and the Contract Certificate is marked as revoked, OR when the SECC or CSMS matches the OEM Provisioning Certificate contained in a CertificateInstallationReq with a CRL and the OEM Provisioning Certificate is marked as revoked. The revocation status can alternatively be obtained through an OCSP responder.
NoCertificateAvailable	If the new certificate cannot be retrieved from secondary actor within the specified timeout
CertChainError	If the ContractSignatureCertChain contained in the CertificateInstallationReq message is not valid.
ContractCancelled	If the EMAID provided by EVCC during CertificateUpdateReq is not accepted by secondary actor.

3.5. BootReasonEnumType

Enumeration

BootReasonEnumType is used by: [bootNotification:BootNotificationRequest](#)

Value	Description
ApplicationReset	The Charging Station rebooted due to an application error.
FirmwareUpdate	The Charging Station rebooted due to a firmware update.
LocalReset	The Charging Station rebooted due to a local reset command.
PowerUp	The Charging Station powered up and registers itself with the CSMS.
RemoteReset	The Charging Station rebooted due to a remote reset command.
ScheduledReset	The Charging Station rebooted due to a scheduled reset command.
Triggered	Requested by the CSMS via a TriggerMessage
Unknown	The boot reason is unknown.
Watchdog	The Charging Station rebooted due to an elapsed watchdog timer.

3.6. CancelReservationStatusEnumType

Enumeration

Status in CancelReservationResponse.

CancelReservationStatusEnumType is used by: [cancelReservation:CancelReservationResponse](#)

Value	Description
Accepted	Reservation for the identifier has been canceled.
Rejected	Reservation could not be canceled, because there is no reservation active for the identifier.

3.7. CertificateActionEnumType

Enumeration

CertificateActionEnumType is used by: [get15118EVCertificate:Get15118EVCertificateRequest](#)

Value	Description
Install	Install the provided certificate.

Value	Description
Update	Update the provided certificate.

3.8. CertificateSignedStatusEnumType

Enumeration

CertificateSignedStatusEnumType is used by: [certificateSigned:CertificateSignedResponse](#)

Value	Description
Accepted	Signed certificate is valid.
Rejected	Signed certificate is invalid.

3.9. CertificateSigningUseEnumType

Enumeration

CertificateSigningUseEnumType is used by: [signCertificate:SignCertificateRequest](#) , [certificateSigned:CertificateSignedRequest](#)

Value	Description
ChargingStationCertificate	Client side certificate used by the Charging Station to connect to the CSMS.
V2GCertificate	Use for certificate for 15118 connections. This means that the certificate should be derived from the V2G root.

3.10. ChangeAvailabilityStatusEnumType

Enumeration

Status returned in response to ChangeAvailabilityRequest.

ChangeAvailabilityStatusEnumType is used by: [changeAvailability:ChangeAvailabilityResponse](#)

Value	Description
Accepted	Request has been accepted and will be executed.
Rejected	Request has not been accepted and will not be executed.
Scheduled	Request has been accepted and will be executed when transaction(s) in progress have finished.

3.11. ChargingLimitSourceEnumType

Enumeration

Enumeration for indicating from which source a charging limit originates.

ChargingLimitSourceEnumType is used by: [notifyChargingLimit:NotifyChargingLimitRequest.ChargingLimitType](#) , [clearedChargingLimit:ClearedChargingLimitRequest](#) , [getChargingProfiles:GetChargingProfilesRequest.ChargingProfileCriterionType](#) , [reportChargingProfiles:ReportChargingProfilesRequest](#)

Value	Description
EMS	Indicates that an Energy Management System has sent a charging limit.
Other	Indicates that an external source, not being an EMS or system operator, has sent a charging limit.
SO	Indicates that a System Operator (DSO or TSO) has sent a charging limit.
CSO	Indicates that the CSO has set this charging profile.

3.12. ChargingProfileKindEnumType

Enumeration

Kind of charging profile.

ChargingProfileKindEnumType is used by: [Common:ChargingProfileType](#)

Value	Description
Absolute	Schedule periods are relative to a fixed point in time defined in the schedule. This requires that <i>startSchedule</i> is set to a starting point in time.
Recurring	The schedule restarts periodically at the first schedule period. To be most useful, this requires that <i>startSchedule</i> is set to a starting point in time.
Relative	Charging schedule periods should start when the EVSE is ready to deliver energy. i.e. when the EV driver is authorized and the EV is connected. When a ChargingProfile is received for a transaction that is already charging, then the charging schedule periods should remain relative to the PowerPathClosed moment. No value for <i>startSchedule</i> should be supplied.

3.13. ChargingProfilePurposeEnumType

Enumeration

Purpose of the charging profile.

ChargingProfilePurposeEnumType is used by: [Common:ChargingProfileType](#) , [clearChargingProfile:ClearChargingProfileRequest.ClearChargingProfileType](#) , [getChargingProfiles:GetChargingProfilesRequest.ChargingProfileCriterionType](#)

Value	Description
ChargingStationExternalConstraints	Additional constraints that will be incorporated into a local power schedule. Only valid for a Charging Station. Therefore evse.Id MUST be 0 in the SetChargingProfileRequest message.
ChargingStationMaxProfile	Configuration for the maximum power or current available for an entire Charging Station.
TxDefaultProfile	Default profile that can be configured in the Charging Station. When a new transaction is started, this profile SHALL be used, unless it was a transaction that was started by a RequestStartTransactionRequest with a ChargingProfile that is accepted by the Charging Station.
TxProfile	Profile with constraints to be imposed by the Charging Station on the current transaction, or on a new transaction when this is started via a RequestStartTransactionRequest with a ChargingProfile. A profile with this purpose SHALL cease to be valid when the transaction terminates.

3.14. ChargingProfileStatusEnumType

Enumeration

Status returned in response to SetChargingProfileRequest.

ChargingProfileStatusEnumType is used by: [setChargingProfile:SetChargingProfileResponse](#)

Value	Description
Accepted	Request has been accepted and will be executed.
Rejected	Request has not been accepted and will not be executed.

3.15. ChargingRateUnitEnumType

Enumeration

Unit in which a charging schedule is defined.

ChargingRateUnitEnumType is used by: [Common:ChargingScheduleType](#) , [Common:CompositeScheduleType](#) ,

[getCompositeSchedule:GetCompositeScheduleRequest](#)

Value	Description
W	Watts (power). This is the TOTAL allowed charging power. If used for AC Charging, the phase current should be calculated via: Current per phase = Power / (Line Voltage * Number of Phases). The "Line Voltage" used in the calculation is not the measured voltage, but the set voltage for the area (hence, 230 or 110 volt). The "Number of Phases" is the numberPhases from the ChargingSchedulePeriod. It is usually more convenient to use this for DC charging. Note that if numberPhases in a ChargingSchedulePeriod is absent, 3 SHALL be assumed.
A	Amperes (current). The amount of Ampere per phase, not the sum of all phases. It is usually more convenient to use this for AC charging.

3.16. ChargingStateEnumType

Enumeration

The state of the charging process.

ChargingStateEnumType is used by: [transactionEvent:TransactionEventRequest.TransactionType](#)

Value	Description
Charging	The contactor of the Connector is closed and energy is flowing to between EVSE and EV.
EVConnected	There is a connection between EV and EVSE, in case the protocol used between EV and the Charging Station can detect a connection, the protocol needs to detect this for the state to become active. The connection can either be wired or wireless.
SuspendedEV	When the EV is connected to the EVSE and the EVSE is offering energy but the EV is not taking any energy.
SuspendedEVSE	When the EV is connected to the EVSE but the EVSE is not offering energy to the EV, e.g. due to a smart charging restriction, local supply power constraints, or when charging has stopped because of the authorization status in the response to a transactionEventRequest indicating that charging is not allowed etc.
Idle	There is no connection between EV and EVSE.

3.17. ClearCacheStatusEnumType

Enumeration

Status returned in response to ClearCacheRequest.

ClearCacheStatusEnumType is used by: [clearCache:ClearCacheResponse](#)

Value	Description
Accepted	Command has been executed.
Rejected	Command has not been executed.

3.18. ClearChargingProfileStatusEnumType

Enumeration

Status returned in response to ClearChargingProfileRequest.

ClearChargingProfileStatusEnumType is used by: [clearChargingProfile:ClearChargingProfileResponse](#)

Value	Description
Accepted	Request has been accepted and will be executed.
Unknown	No Charging Profile(s) were found matching the request.

3.19. ClearMessageStatusEnumType

Enumeration

Result for a ClearDisplayMessageRequest as used in a ClearDisplayMessageResponse.

ClearMessageStatusEnumType is used by: [clearDisplayMessage:ClearDisplayMessageResponse](#)

Value	Description
Accepted	Request successfully executed: message cleared.
Unknown	Given message (based on the id) not known.

3.20. ClearMonitoringStatusEnumType

Enumeration

ClearMonitoringStatusEnumType is used by: [Common:ClearMonitoringResultType](#)

Value	Description
Accepted	Monitor successfully cleared.
Rejected	Clearing of monitor rejected.
NotFound	Monitor Id is not found.

3.21. ComponentCriterionEnumType

Enumeration

ComponentCriterionEnumType is used by: [getReport:GetReportRequest](#)

Value	Description
Active	Components that are active, i.e. having <i>Active</i> = 1
Available	Components that are available, i.e. having <i>Available</i> = 1
Enabled	Components that are enabled, i.e. having <i>Enabled</i> = 1
Problem	Components that reported a problem, i.e. having <i>Problem</i> = 1

3.22. ConnectorEnumType

Enumeration

Allowed values of ConnectorCode.

NOTE

This enumeration does not attempt to include every possible power connector type worldwide as an individual type, but to specifically define those that are known to be in use (or likely to be in use) in the Charging Stations using the OCPP protocol. In particular, many of the very large number of domestic electrical sockets designs in use in many countries are excluded, unless there is evidence that they are or are likely to be approved for use on Charging Stations in some jurisdictions (e.g. as secondary connectors for charging light EVs such as electric scooters). These light connector types can be represented with the enumeration value Other1PhMax16A. Similarly, any single phase connector not otherwise enumerated that is rated for 16A or over should be reported as Other1PhOver16A. All 3 phase connector types not explicitly enumerated should be represented as Other3Ph.

ConnectorEnumType is used by: [reserveNow:ReserveNowRequest](#)

Value	Description
cCCS1	Combined Charging System 1 (captive cabled) a.k.a. Combo 1
cCCS2	Combined Charging System 2 (captive cabled) a.k.a. Combo 2
CG105	JARI G105-1993 (captive cabled) a.k.a. CHAdeMO
cTesla	Tesla Connector (captive cabled)

Value	Description
cType1	IEC62196-2 Type 1 connector (captive cabled) a.k.a. J1772
cType2	IEC62196-2 Type 2 connector (captive cabled) a.k.a. Mennekes connector
s309-1P-16A	16A 1 phase IEC60309 socket
s309-1P-32A	32A 1 phase IEC60309 socket
s309-3P-16A	16A 3 phase IEC60309 socket
s309-3P-32A	32A 3 phase IEC60309 socket
sBS1361	UK domestic socket a.k.a. 13Amp
sCEE-7-7	CEE 7/7 16A socket. May represent 7/4 & 7/5 a.k.a Schuko
sType2	IEC62196-2 Type 2 socket a.k.a. Mennekes connector
sType3	IEC62196-2 Type 2 socket a.k.a. Scame
Other1PhMax16A	Other single phase (domestic) sockets not mentioned above, rated at no more than 16A. CEE7/17, AS3112, NEMA 5-15, NEMA 5-20, JISC8303, TIS166, SI 32, CPCS-CCC, SEV1011, etc.
Other1PhOver16A	Other single phase sockets not mentioned above (over 16A)
Other3Ph	Other 3 phase sockets not mentioned above. NEMA14-30, NEMA14-50.
Pan	Pantograph connector
wInductive	Wireless inductively coupled connection (generic)
wResonant	Wireless resonant coupled connection (generic)
Undetermined	Yet to be determined (e.g. before plugged in)
Unknown	Unknown; not determinable

3.23. ConnectorStatusEnumType

Enumeration

A status can be reported for the Connector of an EVSE of a Charging Station. States considered Operative are: Available, Reserved and Occupied. States considered Inoperative are: Unavailable, Faulted.

ConnectorStatusEnumType is used by: [statusNotification:StatusNotificationRequest](#)

Value	Description
Available	When a Connector becomes available for a new User (Operative)
Occupied	When a Connector becomes occupied, so it is not available for a new EV driver. (Operative)
Reserved	When a Connector becomes reserved as a result of ReserveNow command (Operative)
Unavailable	When a Connector becomes unavailable as the result of a Change Availability command or an event upon which the Charging Station transitions to unavailable at its discretion. Upon receipt of ChangeAvailability message command, the status MAY change immediately or the change MAY be scheduled. When scheduled, StatusNotification SHALL be send when the availability change becomes effective (Inoperative)
Faulted	When a Connector (or the EVSE or the entire Charging Station it belongs to) has reported an error and is not available for energy delivery. (Inoperative).

3.24. CostKindEnumType

Enumeration

CostKindEnumType is used by: [Common:CostType](#)

Value	Description
CarbonDioxideEmission	Absolute value. Carbon Dioxide emissions, in grams per kWh.
RelativePricePercentage	Relative value. Price per kWh, as percentage relative to the maximum price stated in any of all tariffs indicated to the EV.
RenewableGenerationPercentage	Relative value. Percentage of renewable generation within total generation.

3.25. CustomerInformationStatusEnumType

Enumeration

Status in CancelReservationResponse.

CustomerInformationStatusEnumType is used by: [customerInformation:CustomerInformationResponse](#)

Value	Description
Accepted	The Charging Station accepted the message.
Rejected	When the Charging Station is in a state where it cannot process this request.
Invalid	In a request to the Charging Station no reference to a customer is included.

3.26. DataEnumType

Enumeration

DataEnumType is used by: [Common:VariableCharacteristicsType](#)

Value	Description
string	This variable is of the type string.
decimal	This variable is of the type decimal.
integer	This variable is of the type integer.
dateTime	DateTime following the [RFC3339] specification.
boolean	This variable is of the type boolean.
OptionList	Supported/allowed values for a single choice, enumerated, text variable.
SequenceList	Supported/allowed values for an ordered sequence variable.
MemberList	Supported/allowed values for a mathematical set variable.

3.27. DataTransferStatusEnumType

Enumeration

Status in DataTransferResponse.

DataTransferStatusEnumType is used by: [dataTransfer:DataTransferResponse](#)

Value	Description
Accepted	Message has been accepted and the contained request is accepted.
Rejected	Message has been accepted but the contained request is rejected.
UnknownMessageId	Message could not be interpreted due to unknown messageId string.
UnknownVendorId	Message could not be interpreted due to unknown vendorId string.

3.28. DeleteCertificateStatusEnumType

Enumeration

DeleteCertificateStatusEnumType is used by: [deleteCertificate:DeleteCertificateResponse](#)

Value	Description
Accepted	Normal successful completion (no errors).
Failed	The Charging Station either failed to remove the certificate or rejected the request. A Charging Station may reject the request to prevent the deletion of a certificate, if it is the last one from its certificate type.
NotFound	Requested resource not found.

3.29. DisplayMessageStatusEnumType

Enumeration

Result for a SetDisplayMessageRequest as used in a SetDisplayMessageResponse.

DisplayMessageStatusEnumType is used by: [setDisplayMessage:SetDisplayMessageResponse](#)

Value	Description
Accepted	Request to display message accepted.
NotSupportedMessageFormat	None of the formats in the given message are supported.
Rejected	Request cannot be handled.
NotSupportedPriority	The given MessagePriority not supported for displaying messages by Charging Station.
NotSupportedState	The given MessageState not supported for displaying messages by Charging Station.
UnknownTransaction	Given Transaction not known/ongoing.

3.30. EnergyTransferModeEnumType

Enumeration

Enumeration of energy transfer modes.

EnergyTransferModeEnumType is used by: [Common:ChargingNeedsType](#)

Value	Description
DC	DC charging.
AC_single_phase	AC single phase charging according to IEC 62196.
AC_two_phase	AC two phase charging according to IEC 62196.
AC_three_phase	AC three phase charging according to IEC 62196.

3.31. EventNotificationEnumType

Enumeration

Specifies the event notification type of the message.

EventNotificationEnumType is used by: [notifyEvent:NotifyEventRequest.EventDataType](#)

Value	Description
HardWiredNotification	The software implemented by the manufacturer triggered a hardwired notification.
HardWiredMonitor	Triggered by a monitor, which is hardwired by the manufacturer.
PreconfiguredMonitor	Triggered by a monitor, which is preconfigured by the manufacturer.
CustomMonitor	Triggered by a monitor, which is set with the setvariablemonitoringrequest message by the Charging Station Operator.

3.32. EventTriggerEnumType

Enumeration

EventTriggerEnumType is used by: [notifyEvent:NotifyEventRequest.EventDataType](#)

Value	Description
Alerting	Monitored variable has passed an Lower or Upper Threshold
Delta	Delta Monitored Variable value has changed by more than specified amount
Periodic	Periodic Monitored Variable has been sampled for reporting at the specified interval

3.33. FirmwareStatusEnumType

Enumeration

Status of a firmware download.

A value with "Intermediate state" in the description, is an intermediate state, update process is not finished.

A value with "Failure end state" in the description, is an end state, update process has stopped, update failed.

A value with "Successful end state" in the description, is an end state, update process has stopped, update successful.

FirmwareStatusEnumType is used by: [firmwareStatusNotification:FirmwareStatusNotificationRequest](#)

Value	Description
Downloaded	Intermediate state. New firmware has been downloaded by Charging Station.
DownloadFailed	Failure end state. Charging Station failed to download firmware.
Downloading	Intermediate state. Firmware is being downloaded.
DownloadScheduled	Intermediate state. Downloading of new firmware has been scheduled.
DownloadPaused	Intermediate state. Downloading has been paused.
Idle	Charging Station is not performing firmware update related tasks. Status Idle SHALL only be used as in a FirmwareStatusNotificationRequest that was triggered by TriggerMessageRequest.
InstallationFailed	Failure end state. Installation of new firmware has failed.
Installing	Intermediate state. Firmware is being installed.
Installed	Successful end state. New firmware has successfully been installed in Charging Station.
InstallRebooting	Intermediate state. Charging Station is about to reboot to activate new firmware. This status MAY be omitted if a reboot is an integral part of the installation and cannot be reported separately.
InstallScheduled	Intermediate state. Installation of the downloaded firmware is scheduled to take place on installDateTime given in UpdateFirmware request.
InstallVerificationFailed	Failure end state. Verification of the new firmware (e.g. using a checksum or some other means) has failed and installation will not proceed. (Final failure state)
InvalidSignature	Failure end state. The firmware signature is not valid.
SignatureVerified	Intermediate state. Provide signature successfully verified.

3.34. GenericDeviceModelStatusEnumType

Enumeration

GenericDeviceModelStatusEnumType is used by: [getBaseReport:GetBaseReportResponse](#) , [getMonitoringReport:GetMonitoringReportResponse](#) , [getReport:GetReportResponse](#) , [setMonitoringBase:SetMonitoringBaseResponse](#)

Value	Description
Accepted	Request has been accepted and will be executed.
Rejected	Request has not been accepted and will not be executed.
NotSupported	The content of the request message is not supported.
EmptyResultSet	If the combination of received criteria result in an empty result set.

3.35. GenericStatusEnumType

Enumeration

Generic message response status

GenericStatusEnumType is used by: [getCompositeSchedule:GetCompositeScheduleResponse](#) , [notifyEVChargingSchedule:NotifyEVChargingScheduleResponse](#) , [signCertificate:SignCertificateResponse](#) , [setMonitoringLevel:SetMonitoringLevelResponse](#) , [publishFirmware:PublishFirmwareResponse](#)

Value	Description
Accepted	Request has been accepted and will be executed.
Rejected	Request has not been accepted and will not be executed.

3.36. GetCertificateIdUseEnumType

Enumeration

GetCertificateIdUseEnumType is used by: [Common:CertificateHashDataChainType](#) , [getInstalledCertificates:GetInstalledCertificatesRequest](#)

Value	Description
V2GRootCertificate	Use for certificate of the V2G Root.
MORootCertificate	Use for certificate from an eMobility Service provider. To support PnC charging with contracts from service providers that not derived their certificates from the V2G root.
CSMSRootCertificate	Root certificate for verification of the CSMS certificate.
V2GCertificateChain	ISO 15118 V2G certificate chain (excluding the V2GRootCertificate).
ManufacturerRootCertificate	Root certificate for verification of the Manufacturer certificate.

3.37. GetCertificateStatusEnumType

Enumeration

GetCertificateStatusEnumType is used by: [getCertificateStatus:GetCertificateStatusResponse](#)

Value	Description
Accepted	Successfully retrieved the OCSP certificate status.
Failed	Failed to retrieve the OCSP certificate status.

3.38. GetChargingProfileStatusEnumType

Enumeration

GetChargingProfileStatusEnumType is used by: [getChargingProfiles:GetChargingProfilesResponse](#)

Value	Description
Accepted	Normal successful completion (no errors).
NoProfiles	No ChargingProfiles found that match the information in the GetChargingProfilesRequest .

3.39. GetDisplayMessagesStatusEnumType

Enumeration

GetDisplayMessagesStatusEnumType is used by: [getDisplayMessages:GetDisplayMessagesResponse](#)

Value	Description
Accepted	Request accepted, there are Display Messages found that match all the requested criteria. The Charging Station will send NotifyDisplayMessagesRequest messages to report the requested Display Messages.
Unknown	No messages found that match the given criteria.

3.40. GetInstalledCertificateStatusEnumType

Enumeration

GetInstalledCertificateStatusEnumType is used by: [getInstalledCertificateIds:GetInstalledCertificateIdsResponse](#)

Value	Description
Accepted	Normal successful completion (no errors).
NotFound	Requested resource not found.

3.41. GetVariableStatusEnumType

Enumeration

GetVariableStatusEnumType is used by: [getVariables:GetVariablesResponse.GetVariableResultType](#)

Value	Description
Accepted	Variable successfully set.
Rejected	Request is rejected.
UnknownComponent	Component is not known.
UnknownVariable	Variable is not known.
NotSupportedAttributeType	The AttributeType is not supported.

3.42. HashAlgorithmEnumType

Enumeration

HashAlgorithmEnumType is used by: [Common:CertificateHashDataType](#) , [Common:OCSPRequestDataType](#)

Value	Description
SHA256	SHA-256 hash algorithm.
SHA384	SHA-384 hash algorithm.
SHA512	SHA-512 hash algorithm.

3.43. IdTokenType

Enumeration

Allowable values of the **IdTokenType** field.

IdTokenType is used by: [Common:IdTokenType](#)

Value	Description
Central	A centrally, in the CSMS (or other server) generated id (for example used for a remotely started transaction that is activated by SMS). No format defined, might be a UUID.
eMAID	Electro-mobility account id as defined in ISO 15118
ISO14443	ISO 14443 UID of RFID card. It is represented as an array of 4 or 7 bytes in hexadecimal representation.
ISO15693	ISO 15693 UID of RFID card. It is represented as an array of 8 bytes in hexadecimal representation.

Value	Description
KeyCode	User use a private key-code to authorize a charging transaction. For example: Pin-code.
Local	A locally generated id (e.g. internal id created by the Charging Station). No format defined, might be a UUID
MacAddress	
NoAuthorization	Transaction is started and no authorization possible. Charging Station only has a start button or mechanical key etc. IdToken field SHALL be left empty.

3.44. InstallCertificateStatusEnumType

Enumeration

InstallCertificateStatusEnumType is used by: [installCertificate:InstallCertificateResponse](#)

Value	Description
Accepted	The installation of the certificate succeeded.
Rejected	The certificate is invalid and/or incorrect OR the CSO tries to install more certificates than allowed.
Failed	The certificate is valid and correct, but there is another reason the installation did not succeed.

3.45. InstallCertificateUseEnumType

Enumeration

InstallCertificateUseEnumType is used by: [installCertificate:InstallCertificateRequest](#)

Value	Description
V2GRootCertificate	Use for certificate of the V2G Root, a V2G Charging Station Certificate MUST be derived from one of the installed V2GRootCertificate certificates.
MORootCertificate	Use for certificate from an eMobility Service provider. To support PnC charging with contracts from service providers that not derived their certificates from the V2G root.
CSMSRootCertificate	Root certificate, used by the CA to sign the CSMS and Charging Station certificate.
ManufacturerRootCertificate	Root certificate for verification of the Manufacturer certificate.

3.46. Iso15118EVCertificateStatusEnumType

Enumeration

Iso15118EVCertificateStatusEnumType is used by: [get15118EVCertificate:Get15118EVCertificateResponse](#)

Value	Description
Accepted	exiResponse included. This is no indication whether the update was successful, just that the message was processed properly.
Failed	Processing of the message was not successful, no exiResponse included.

3.47. LocationEnumType

Enumeration

Allowable values of the optional "location" field of a value element.

LocationEnumType is used by: [Common:SampledValueType](#)

Value	Description
Body	Measurement inside body of Charging Station (e.g. Temperature).
Cable	Measurement taken from cable between EV and Charging Station.

Value	Description
EV	Measurement taken by EV.
Inlet	For the Charging Station (<code>evselId</code> = 0): measurement at network ("grid") inlet connection. For measurements with <code>evselId</code> > 0, these are measurements taken at the EVSE inlet (This can be useful for a DC charger).
Outlet	Measurement at a Connector. Default value.

3.48. LogEnumType

Enumeration

LogEnumType is used by: [getLog:GetLogRequest](#)

Value	Description
DiagnosticsLog	This contains the field definition of a diagnostics log file
SecurityLog	Sent by the CSMS to the Charging Station to request that the Charging Station uploads the security log.

3.49. LogStatusEnumType

Enumeration

Generic message response status

LogStatusEnumType is used by: [getLog:GetLogResponse](#)

Value	Description
Accepted	Accepted this log upload. This does not mean the log file is uploaded successfully, the Charging Station will now start the log file upload.
Rejected	Log update request rejected.
AcceptedCanceled	Accepted this log upload, but in doing this has canceled an ongoing log file upload.

3.50. MeasurandEnumType

Enumeration

Allowable values of the optional "measurand" field of a Value element, as used in [MeterValuesRequest](#) and [TransactionEventRequest](#) with eventTypes `Started`, `Ended` and `Updated`. Default value of "measurand" is always "Energy.Active.Import.Register".

Note 1: Two measurands (Current.Offered and Power.Offered) are available that are strictly speaking no measured values. They indicate the maximum amount of current/power that is being offered to the EV and are intended for use in smart charging applications.

Note 2: Import is energy flow from the Grid to the Charging Station, EV or other load. Export is energy flow from the EV to the Charging Station and/or from the Charging Station to the Grid. Except in the case of a meter replacement, all "Register" values relating to a single charging transaction, or a non-transactional consumer (e.g. Charging Station internal power supply, overall supply) MUST be monotonically increasing in time.

Note 3: The actual quantity of energy corresponding to a reported ".Register" value is computed as the register value in question minus the register value recorded/reported at the start of the transaction or other relevant starting reference point in time. For improved auditability, ".Register" values SHOULD be reported exactly as they are directly read from a non-volatile register in the electrical metering hardware, and SHOULD NOT be re-based to zero at the start of transactions. This allows any "missing energy" between sequential transactions, due to hardware fault, meter replacement, mis-wiring, fraud, etc. to be identified, by allowing the CSMS to confirm that the starting register value of any transaction is identical to the finishing register value of the preceding transaction on the same connector.

MeasurandEnumType is used by: [Common:SampledValueType](#)

Value	Description
Current.Export	Instantaneous current flow from EV

Value	Description
Current.Import	Instantaneous current flow to EV
Current.Offered	Maximum current offered to EV
Energy.Active.Export.Register	Numerical value read from the "active electrical energy" (Wh or kWh) register of the (most authoritative) electrical meter measuring energy exported (to the grid).
Energy.Active.Import.Register	Numerical value read from the "active electrical energy" (Wh or kWh) register of the (most authoritative) electrical meter measuring energy imported (from the grid supply).
Energy.Reactive.Export.Register	Numerical value read from the "reactive electrical energy" (varh or kvarh) register of the (most authoritative) electrical meter measuring energy exported (to the grid).
Energy.Reactive.Import.Register	Numerical value read from the "reactive electrical energy" (varh or kvarh) register of the (most authoritative) electrical meter measuring energy imported (from the grid supply).
Energy.Active.Export.Interval	Absolute amount of "active electrical energy" (Wh or kWh) exported (to the grid) during an associated time "interval", specified by a Metervalues ReadingContext, and applicable interval duration configuration values (in seconds) for ClockAlignedDataInterval and TxnMeterValueSampleInterval.
Energy.Active.Import.Interval	Absolute amount of "active electrical energy" (Wh or kWh) imported (from the grid supply) during an associated time "interval", specified by a Metervalues ReadingContext, and applicable interval duration configuration values (in seconds) for ClockAlignedDataInterval and TxnMeterValueSampleInterval.
Energy.Active.Net	Numerical value read from the "net active electrical energy" (Wh or kWh) register.
Energy.Reactive.Export.Interval	Absolute amount of "reactive electrical energy" (varh or kvarh) exported (to the grid) during an associated time "interval", specified by a Metervalues ReadingContext, and applicable interval duration configuration values (in seconds) for ClockAlignedDataInterval and TxnMeterValueSampleInterval.
Energy.Reactive.Import.Interval	Absolute amount of "reactive electrical energy" (varh or kvarh) imported (from the grid supply) during an associated time "interval", specified by a Metervalues ReadingContext, and applicable interval duration configuration values (in seconds) for ClockAlignedDataInterval and TxnMeterValueSampleInterval.
Energy.Reactive.Net	Numerical value read from the "net reactive electrical energy" (varh or kvarh) register.
Energy.Apparent.Net	Numerical value read from the "apparent electrical energy" (VAh or kVAh) register.
Energy.Apparent.Import	Numerical value read from the "apparent electrical import energy" (VAh or kVAh) register.
Energy.Apparent.Export	Numerical value read from the "apparent electrical export energy" (VAh or kVAh) register.
Frequency	Instantaneous reading of powerline frequency
Power.Active.Export	Instantaneous active power exported by EV. (W or kW)
Power.Active.Import	Instantaneous active power imported by EV. (W or kW)
Power.Factor	Instantaneous power factor of total energy flow
Power.Offered	Maximum power offered to EV
Power.Reactive.Export	Instantaneous reactive power exported by EV. (var or kvar)
Power.Reactive.Import	Instantaneous reactive power imported by EV. (var or kvar)
SoC	State of charge of charging vehicle in percentage
Voltage	Instantaneous DC or AC RMS supply voltage. For <i>location</i> = Inlet and <i>evseld</i> = 0: voltage at charging station grid connection. For <i>location</i> = Outlet and <i>evseld</i> > 0: voltage at EVSE outlet towards the EV.

3.51. MessageFormatEnumType

Enumeration

Format of a message to be displayed on the display of the Charging Station.

MessageFormatEnumType is used by: [Common:MessageContentType](#)

Value	Description
ASCII	Message content is ASCII formatted, only printable ASCII allowed.
HTML	Message content is HTML formatted.

Value	Description
URI	Message content is URI that Charging Station should download and use to display. for example a HTML page to be shown in a web-browser.
UTF8	Message content is UTF-8 formatted.

3.52. MessagePriorityEnumType

Enumeration

Priority with which a message should be displayed on a Charging Station.

MessagePriorityEnumType is used by: [Common:MessageInfoType](#) , [getDisplayMessages:GetDisplayMessagesRequest](#)

Value	Description
AlwaysFront	Show this message always in front. Highest priority, don't cycle with other messages. When a newer message with this MessagePriority is received, this message is replaced. No Charging Station own message may override this message.
InFront	Show this message in front of the normal cycle of messages. When more messages with this priority are to be shown, they SHALL be cycled.
NormalCycle	Show this message in the cycle of messages.

3.53. MessageStateEnumType

Enumeration

State of the Charging Station during which a message SHALL be displayed.

MessageStateEnumType is used by: [Common:MessageInfoType](#) , [getDisplayMessages:GetDisplayMessagesRequest](#)

Value	Description
Charging	Message only to be shown while the Charging Station is charging.
Faulted	Message only to be shown while the Charging Station is in faulted state.
Idle	Message only to be shown while the Charging Station is idle (not charging).
Unavailable	Message only to be shown while the Charging Station is in unavailable state.

3.54. MessageTriggerEnumType

Enumeration

Type of request to be triggered by trigger messages.

MessageTriggerEnumType is used by: [triggerMessage:TriggerMessageRequest](#)

Value	Description
BootNotification	To trigger BootNotification.
LogStatusNotification	To trigger LogStatusNotification.
FirmwareStatusNotification	To trigger FirmwareStatusNotification.
Heartbeat	To trigger Heartbeat.
MeterValues	To trigger MeterValues.
SignChargingStationCertificate	To trigger a SignCertificate with certificateType: ChargingStationCertificate.
SignV2GCertificate	To trigger a SignCertificate with certificateType: V2GCertificate
StatusNotification	To trigger StatusNotification.
TransactionEvent	To trigger TransactionEvent.

Value	Description
SignCombinedCertificate	To trigger a SignCertificate with certificateType: ChargingStationCertificate AND V2GCertificate
PublishFirmwareStatusNotification	To trigger PublishFirmwareStatusNotification .

3.55. MonitorEnumType

Enumeration

MonitorEnumType is used by: [Common:VariableMonitoringType](#),
[setVariableMonitoring:SetVariableMonitoringRequest.SetMonitoringDataType](#),
[setVariableMonitoring:SetVariableMonitoringResponse.SetMonitoringResultType](#)

Value	Description
UpperThreshold	Triggers an event notice when the actual value of the Variable rises above <i>monitorValue</i>
LowerThreshold	Triggers an event notice when the actual value of the Variable drops below <i>monitorValue</i> .
Delta	Triggers an event notice when the actual value has changed more than plus or minus <i>monitorValue</i> since the time that this monitor was set or since the last time this event notice was sent, whichever was last. For variables that are not numeric, like boolean, string or enumerations, a monitor of type Delta will trigger an event notice whenever the variable changes, regardless of the value of <i>monitorValue</i> .
Periodic	Triggers an event notice every <i>monitorValue</i> seconds interval, starting from the time that this monitor was set.
PeriodicClockAligned	Triggers an event notice every <i>monitorValue</i> seconds interval, starting from the nearest clock-aligned interval after this monitor was set. For example, a <i>monitorValue</i> of 900 will trigger event notices at 0, 15, 30 and 45 minutes after the hour, every hour.

3.56. MonitoringBaseEnumType

Enumeration

MonitoringBaseEnumType is used by: [setMonitoringBase:SetMonitoringBaseRequest](#)

Value	Description
All	Activate all pre-configured monitors.
FactoryDefault	Activate the default monitoring settings as recommended by the manufacturer. This is a subset of all pre-configured monitors.
HardWiredOnly	Clears all custom monitors and disables all pre-configured monitors.

3.57. MonitoringCriterionEnumType

Enumeration

MonitoringCriterionEnumType is used by: [getMonitoringReport:GetMonitoringReportRequest](#)

Value	Description
ThresholdMonitoring	Report variables and components with a monitor of type UpperThreshold or LowerThreshold.
DeltaMonitoring	Report variables and components with a monitor of type Delta.
PeriodicMonitoring	Report variables and components with a monitor of type Periodic or PeriodicClockAligned.

3.58. MutabilityEnumType

Enumeration

MutabilityEnumType is used by: [Common:VariableAttributeType](#)

Value	Description
ReadOnly	This variable is read-only.
WriteOnly	This variable is write-only.
ReadWrite	This variable is read-write.

3.59. NotifyEVChargingNeedsStatusEnumType

Enumeration

NotifyEVChargingNeedsStatusEnumType is used by: [notifyEVChargingNeeds:NotifyEVChargingNeedsResponse](#)

Value	Description
Accepted	A schedule will be provided momentarily.
Rejected	Service not available.
Processing	The CSMS is gathering information to provide a schedule.

3.60. OCPPInterfaceEnumType

Enumeration

Enumeration of network interfaces.

OCPPInterfaceEnumType is used by: [setNetworkProfile:SetNetworkProfileRequest.NetworkConnectionProfileType](#)

Value	Description
Wired0	Use wired connection 0
Wired1	Use wired connection 1
Wired2	Use wired connection 2
Wired3	Use wired connection 3
Wireless0	Use wireless connection 0
Wireless1	Use wireless connection 1
Wireless2	Use wireless connection 2
Wireless3	Use wireless connection 3

3.61. OCPPTransportEnumType

Enumeration

Enumeration of OCPP transport mechanisms. SOAP is currently not a valid value for OCPP 2.0.

OCPPTransportEnumType is used by: [setNetworkProfile:SetNetworkProfileRequest.NetworkConnectionProfileType](#)

Value	Description
JSON	Use JSON over WebSockets for transport of OCPP PDU's
SOAP	Use SOAP for transport of OCPP PDU's

3.62. OCPPVersionEnumType

Enumeration

Enumeration of OCPP versions.

OCPPVersionEnumType is used by: [setNetworkProfile:SetNetworkProfileRequest.NetworkConnectionProfileType](#)

Value	Description
OCPP12	OCPP version 1.2
OCPP15	OCPP version 1.5
OCPP16	OCPP version 1.6
OCPP20	OCPP version 2.0 The OCPP 2.0 release of OCPP has been deprecated, so this value OCPP20 must now be used for OCPP 2.0.1 implementations in the NetworkConnectionProfile. Note that OCPP 2.0.1 does have its own Websocket subprotocol name: ocpp2.0.1.

3.63. OperationalStatusEnumType

Enumeration

Requested availability change.

OperationalStatusEnumType is used by: [changeAvailability:ChangeAvailabilityRequest](#)

Value	Description
Inoperative	Charging Station is not available for charging.
Operative	Charging Station is available for charging.

3.64. PhaseEnumType

Enumeration

Phase specifies how a measured value is to be interpreted. Please note that not all values of Phase are applicable to all Measurands.

PhaseEnumType is used by: [Common:SampledValueType](#)

Value	Description
L1	Measured on L1
L2	Measured on L2
L3	Measured on L3
N	Measured on Neutral
L1-N	Measured on L1 with respect to Neutral conductor
L2-N	Measured on L2 with respect to Neutral conductor
L3-N	Measured on L3 with respect to Neutral conductor
L1-L2	Measured between L1 and L2
L2-L3	Measured between L2 and L3
L3-L1	Measured between L3 and L1

3.65. PublishFirmwareStatusEnumType

Enumeration

Status for when publishing a Firmware.

PublishFirmwareStatusEnumType is used by: [publishFirmwareStatusNotification:PublishFirmwareStatusNotificationRequest](#)

Value	Description
Idle	
DownloadScheduled	Intermediate state. Downloading of new firmware has been scheduled.
Downloading	Intermediate state. Firmware is being downloaded.
Downloaded	Intermediate state. New firmware has been downloaded by Charging Station.

Value	Description
Published	The firmware has been successfully published.
DownloadFailed	Failure end state. Charging Station failed to download firmware.
DownloadPaused	Intermediate state. Downloading has been paused.
InvalidChecksum	Failure end state. The firmware checksum is not matching.
ChecksumVerified	Intermediate state. The Firmware checksum is successfully verified.
PublishFailed	Publishing the new firmware has failed.

3.66. ReadingContextEnumType

Enumeration

Values of the context field.

ReadingContextEnumType is used by: [Common:SampledValueType](#)

Value	Description
Interruption.Begin	Value taken at start of interruption.
Interruption.End	Value taken when resuming after interruption.
Other	Value for any other situations.
Sample.Clock	Value taken at clock aligned interval.
Sample.Periodic	Value taken as periodic sample relative to start time of transaction.
Transaction.Begin	Value taken at start of transaction.
Transaction.End	Value taken at end of transaction.
Trigger	Value taken in response to TriggerMessageRequest.

3.67. ReasonEnumType

Enumeration

Reason for stopping a transaction.

ReasonEnumType is used by: [transactionEvent:TransactionEventRequest.TransactionType](#)

Value	Description
DeAuthorized	The transaction was stopped because of the authorization status in the response to a transactionEventRequest.
EmergencyStop	Emergency stop button was used.
EnergyLimitReached	EV charging session reached a locally enforced maximum energy transfer limit
EVDisconnected	Disconnecting of cable, vehicle moved away from inductive charge unit.
GroundFault	A GroundFault has occurred
ImmediateReset	A Reset(Immediate) command was received.
Local	Stopped locally on request of the EV Driver at the Charging Station. This is a regular termination of a transaction. Examples: presenting an IdToken tag, pressing a button to stop.
LocalOutOfCredit	A local credit limit enforced through the Charging Station has been exceeded.
MasterPass	The transaction was stopped using a token with a MasterPassGroupId.
Other	Any other reason.
OvercurrentFault	A larger than intended electric current has occurred
PowerLoss	Complete loss of power.
PowerQuality	Quality of power too low, e.g. voltage too low/high, phase imbalance, etc.
Reboot	A locally initiated reset/reboot occurred. (for instance watchdog kicked in)
Remote	Stopped remotely on request of the CSMS. This is a regular termination of a transaction. Examples: termination using a smartphone app, exceeding a (non local) prepaid credit.
SOCLimitReached	Electric vehicle has reported reaching a locally enforced maximum battery State of Charge (SOC)

Value	Description
StoppedByEV	The transaction was stopped by the EV
TimeLimitReached	EV charging session reached a locally enforced time limit
Timeout	EV not connected within timeout

3.68. RecurrencyKindEnumType

Enumeration

RecurrencyKindEnumType is used by: [Common:ChargingProfileType](#)

Value	Description
Daily	The schedule restarts at the beginning of the next day.
Weekly	The schedule restarts at the beginning of the next week (defined as Monday morning)

3.69. RegistrationStatusEnumType

Enumeration

Result of registration in response to BootNotificationRequest.

RegistrationStatusEnumType is used by: [bootNotification:BootNotificationResponse](#)

Value	Description
Accepted	Charging Station is accepted by the CSMS.
Pending	CSMS is not yet ready to accept the Charging Station. CSMS may send messages to retrieve information or prepare the Charging Station.
Rejected	Charging Station is not accepted by CSMS. This may happen when the Charging Station id is not known by CSMS.

3.70. ReportBaseEnumType

Enumeration

ReportBaseEnumType is used by: [getBaseReport:GetBaseReportRequest](#)

Value	Description
ConfigurationInventory	Required. A (configuration) report that lists all Components/Variables that can be set by the operator.
FullInventory	Required. A (full) report that lists everything except monitoring settings.

Value	Description
SummaryInventory	<p>Optional. A (summary) report that lists Components/Variables relating to the Charging Station's current charging availability, and to any existing problem conditions.</p> <p>For the Charging Station Component:</p> <ul style="list-style-type: none"> - AvailabilityState. <p>For each EVSE Component:</p> <ul style="list-style-type: none"> - AvailabilityState. <p>For each Connector Component:</p> <ul style="list-style-type: none"> - AvailabilityState (if known and different from EVSE). <p>For all Components in an abnormal State:</p> <ul style="list-style-type: none"> - Active (Problem, Tripped, Overload, Fallback) variables. - Any other diagnostically relevant Variables of the Components. - Include TechCode and TechInfo where available. <p>All monitored Component.Variables in Critical or Alert state shall also be included.</p> <ul style="list-style-type: none"> - Charging Stations that do not have Monitoring implemented are NOT REQUIRED to include Connector Availability, monitoring alerts, and MAY limit problem reporting detail to just the active Problem boolean Variable.

3.71. RequestStartStopStatusEnumType

Enumeration

The result of a RequestStartTransactionRequest or RequestStopTransactionRequest.

RequestStartStopStatusEnumType is used by: [requestStartTransaction:RequestStartTransactionResponse](#), [requestStopTransaction:RequestStopTransactionResponse](#)

Value	Description
Accepted	Command will be executed.
Rejected	Command will not be executed.

3.72. ReservationUpdateStatusEnumType

Enumeration

ReservationUpdateStatusEnumType is used by: [reservationStatusUpdate:ReservationStatusUpdateRequest](#)

Value	Description
Expired	The reservation is expired.
Removed	The reservation is removed.

3.73. ReserveNowStatusEnumType

Enumeration

Status in ReserveNowResponse.

ReserveNowStatusEnumType is used by: [reserveNow:ReserveNowResponse](#)

Value	Description
Accepted	Reservation has been made.
Faulted	Reservation has not been made, because evse, connectors or specified connector are in a faulted state.
Occupied	Reservation has not been made. The evse or the specified connector is occupied.
Rejected	Reservation has not been made. Charging Station is not configured to accept reservations.

Value	Description
Unavailable	Reservation has not been made, because evse, connectors or specified connector are in an unavailable state.

3.74. ResetEnumType

Enumeration

Type of reset requested.

ResetEnumType is used by: [reset:ResetRequest](#)

Value	Description
Immediate	Immediate reset of the Charging Station.
OnIdle	Delay reset until no more transactions are active.

3.75. ResetStatusEnumType

Enumeration

Result of ResetRequest.

ResetStatusEnumType is used by: [reset:ResetResponse](#)

Value	Description
Accepted	Command will be executed.
Rejected	Command will not be executed.
Scheduled	Reset command is scheduled, Charging Station is busy with a process that cannot be interrupted at the moment. Reset will be executed when process is finished.

3.76. SendLocalListStatusEnumType

Enumeration

Type of update for SendLocalListRequest.

SendLocalListStatusEnumType is used by: [sendLocalList:SendLocalListResponse](#)

Value	Description
Accepted	Local Authorization List successfully updated.
Failed	Failed to update the Local Authorization List.
VersionMismatch	Version number in the request for a differential update is less or equal than version number of current list.

3.77. SetMonitoringStatusEnumType

Enumeration

SetMonitoringStatusEnumType is used by: [setVariableMonitoring:SetVariableMonitoringResponse.SetMonitoringResultType](#)

Value	Description
Accepted	Monitor successfully set.
UnknownComponent	Component is not known.
UnknownVariable	Variable is not known.
UnsupportedMonitorType	Requested monitor type is not supported.

Value	Description
Rejected	Request is rejected.
Duplicate	A monitor already exists for the given type/severity combination.

3.78. SetNetworkProfileStatusEnumType

Enumeration

Possible values of SetNetworkProfileStatus as used in SetNetworkProfileResponse.

SetNetworkProfileStatusEnumType is used by: [setNetworkProfile: SetNetworkProfileResponse](#)

Value	Description
Accepted	Setting new data successful
Rejected	Setting new data rejected
Failed	Setting new data failed

3.79. SetVariableStatusEnumType

Enumeration

SetVariableStatusEnumType is used by: [setVariables: SetVariablesResponse.SetVariableResultType](#)

Value	Description
Accepted	Variable successfully set.
Rejected	Request is rejected.
UnknownComponent	Component is not known.
UnknownVariable	Variable is not known.
NotSupportedAttributeType	The AttributeType is not supported.
RebootRequired	A reboot is required.

3.80. TransactionEventEnumType

Enumeration

TransactionEventEnumType is used by: [transactionEvent: TransactionEventRequest](#)

Value	Description
Ended	Last event of a transaction
Started	First event of a transaction.
Updated	Transaction event in between 'Started' and 'Ended'.

3.81. TriggerMessageStatusEnumType

Enumeration

Status in TriggerMessageResponse.

TriggerMessageStatusEnumType is used by: [triggerMessage: TriggerMessageResponse](#)

Value	Description
Accepted	Requested message will be sent.
Rejected	Requested message will not be sent.

Value	Description
NotImplemented	Requested message cannot be sent because it is either not implemented or unknown.

3.82. TriggerReasonEnumType

Enumeration

Reason that triggered a transactionEventRequest.

TriggerReasonEnumType is used by: [transactionEvent:TransactionEventRequest](#)

Value	Description
Authorized	Charging is authorized, by any means. Might be an RFID, or other authorization means.
CablePluggedIn	Cable is plugged in and EVDetected.
ChargingRateChanged	Rate of charging changed by more than <i>LimitChangeSignificance</i> .
ChargingStateChanged	Charging State changed.
Deauthorized	The transaction was stopped because of the authorization status in the response to a transactionEventRequest.
EnergyLimitReached	Maximum energy of charging reached. For example: in a pre-paid charging solution
EVCommunicationLost	Communication with EV lost, for example: cable disconnected.
EVConnectTimeout	EV not connected before the connection is timed out.
MeterValueClock	Needed to send a clock aligned meter value
MeterValuePeriodic	Needed to send a periodic meter value
TimeLimitReached	Maximum time of charging reached. For example: in a pre-paid charging solution
Trigger	Requested by the CSMS via a TriggerMessageRequest.
UnlockCommand	CSMS sent an Unlock Connector command.
StopAuthorized	An EV Driver has been authorized to stop charging. For example: By swiping an RFID card.
EVDeparted	EV departed. For example: When a departing EV triggers a parking bay detector.
EVDetected	EV detected. For example: When an arriving EV triggers a parking bay detector.
RemoteStop	A RequestStopTransactionRequest has been sent.
RemoteStart	A RequestStartTransactionRequest has been sent.
AbnormalCondition	An Abnormal Error or Fault Condition has occurred.
SignedDataReceived	Signed data is received from the energy meter.
ResetCommand	CSMS sent a Reset Charging Station command.

3.83. UnlockStatusEnumType

Enumeration

Status in response to UnlockConnectorRequest.

UnlockStatusEnumType is used by: [unlockConnector:UnlockConnectorResponse](#)

Value	Description
Unlocked	Connector has successfully been unlocked.
UnlockFailed	Failed to unlock the connector.
OngoingAuthorizedTransaction	The connector is not unlocked, because there is still an authorized transaction ongoing.
UnknownConnector	The specified connector is not known by the Charging Station.

3.84. UnpublishFirmwareStatusEnumType

Enumeration

Status for when publishing a Firmware.

UnpublishFirmwareStatusEnumType is used by: [unpublishFirmware:UnpublishFirmwareResponse](#)

Value	Description
DownloadOngoing	Intermediate state. Firmware is being downloaded.
NoFirmware	There is no published file.
Unpublished	Successful end state. Firmware file no longer being published.

3.85. UpdateEnumType

Enumeration

UpdateEnumType is used by: [sendLocalList:SendLocalListRequest](#)

Value	Description
Differential	Indicates that the current Local Authorization List must be updated with the values in this message.
Full	Indicates that the current Local Authorization List must be replaced by the values in this message.

3.86. UpdateFirmwareStatusEnumType

Enumeration

Generic message response status

UpdateFirmwareStatusEnumType is used by: [updateFirmware:UpdateFirmwareResponse](#)

Value	Description
Accepted	Accepted this firmware update request. This does not mean the firmware update is successful, the Charging Station will now start the firmware update process.
Rejected	Firmware update request rejected.
AcceptedCanceled	Accepted this firmware update request, but in doing this has canceled an ongoing firmware update.
InvalidCertificate	The certificate is invalid.
RevokedCertificate	Failure end state. The Firmware Signing certificate has been revoked.

3.87. UploadLogStatusEnumType

Enumeration

UploadLogStatusEnumType is used by: [logStatusNotification:LogStatusNotificationRequest](#)

Value	Description
BadMessage	A badly formatted packet or other protocol incompatibility was detected.
Idle	The Charging Station is not uploading a log file. Idle SHALL only be used when the message was triggered by a TriggerMessageRequest.
NotSupportedOperation	The server does not support the operation
PermissionDenied	Insufficient permissions to perform the operation.
Uploaded	File has been uploaded successfully.
UploadFailure	Failed to upload the requested file.
Uploading	File is being uploaded.
AcceptedCanceled	On-going log upload is canceled and new request to upload log has been accepted.

3.88. VPNEnumType

Enumeration

Enumeration of VPN Types.

VPNEnumType is used by: [setNetworkProfile:SetNetworkProfileRequest.VPNTYPE](#)

Value	Description
IKEv2	IKEv2 VPN
IPSec	IPSec VPN
L2TP	L2TP VPN
PPTP	PPTP VPN

Referenced Components and Variables

1. Controller Components

This section gives an overview of the 'Controller' components, which are introduced in OCPP 2.0. A controller component can be recognized by the 'Ctrlr' suffix and is responsible for the configuration of a certain functionality. Most of the '[Referenced](#)' components that are described in this document, are 'Controller' components.

The table below contains a summary of all Controller components, for more details, please refer to Part 2 - Appendices.

Controller Component	Description
AlignedDataCtrlr	Responsible for configuration relating to the reporting of clock-aligned meter data.
AuthCacheCtrlr	Responsible for configuration relating to the use of a local cache for authorization for Charging Station use.
AuthCtrlr	Responsible for configuration relating to the use of authorization for Charging Station use.
CHAdeMOCtrlr	Responsible for configuration relating to the CHAdeMO controller
ClockCtrlr	Provides a means to configure management of time tracking by Charging Station.
CustomizationCtrlr	Responsible for configuration relating to custom vendor-specific implementations, using the DataTransfer message and CustomData extensions.
DeviceDataCtrlr	Responsible for configuration relating to the exchange and storage of Charging Station device model data.
DisplayMessageCtrlr	Responsible for configuration relating to the display of messages to Charging Station users.
ISO15118Ctrlr	Responsible for configuration relating to the ISO 15118 controller
LocalAuthListCtrlr	Responsible for configuration relating to the use of local authorization lists for Charging Station use.
MonitoringCtrlr	Responsible for configuration relating to the exchange of monitoring event data.
OCPPCommCtrlr	Responsible for configuration relating to information exchange between Charging Station and CSMS.
ReservationCtrlr	Responsible for configuration relating to reservations.
SampledDataCtrlr	Responsible for configuration relating to the reporting of sampled meter data.
SecurityCtrlr	Responsible for configuration relating to security of communications between Charging Station and CSMS.
SmartChargingCtrlr	Responsible for configuration relating to Smart Charging.
TariffCostCtrlr	Responsible for configuration relating to tariff and cost display.
TxCtrlr	Responsible for configuration relating to transaction characteristics and behaviour.

Every Controller component has an 'Enabled' variable. This variable can be used to enable/disable a certain functionality. Any data in the charging station is not part of the controller component, so when disabling a functionality, any relating data stored in the Charging Station will not be changed or removed.

For example: if ReservationCtrlr is disabled when there is an active reservation, the EVSE will become available, but the reservation entries will still be there – they are just not used. If afterwards ReservationCtrlr is enabled again, the reservation entries will become active again as long as they have not expired and no transaction is in progress. If a transaction has started in the mean time, that transaction remains active. The reservation is then considered expired.

2. Referenced Components and Variables

Below follows a list of all Component Variable combinations with a role standardized in this specification.

These Configuration Variables replace the Configuration Keys from OCPP 1.x

The list is split by functionality: [General](#), [Security](#), [Authorization](#), [Local Authorization List Management related](#), [Authorization Cache](#), [Transaction](#), [Metering](#), [Reservation](#), [Smart Charging](#), [Tariff & Cost](#), [Diagnostics](#), [Display Message](#) and [Charging Infrastructure](#) related.

A required Configuration Variable mentioned under a particular function block only has to be supported by the Charging Station if it supports that functional block.

Please see chapter 4 in "Part 1 - Architecture & Topology" about the addressing of Components and Variables in the Device Model.

Requirements for all the Configuration Variables in this document:

- All variables that are writable SHALL have the VariableAttribute field: `persistence = true`, and SHALL thus be stored in a persistent way.
- Any fields not defined SHALL be left empty.
- Any field marked with a * (Asterisk) can be of any possible value.
- When the AttributeType is NOT given, the CSMS and Charging Station SHALL assume the AttributeType to be Actual.

NOTE

See '*OCPP 2.0 Part 4 - JSON over Websockets implementation guide*' for a number of Configuration Variables that are specific to controlling the JSON/Websocket behavior.

2.1. General

2.1.1. ActiveNetworkProfile

Required	no		
Component	componentName	OCPPCommCtrlr	
Variable	variableName	ActiveNetworkProfile	
	variableAttributes	mutability	ReadOnly
	variableCharacteristics	dataType	integer
Description	Indicates the configuration profile the station uses at that moment to connect to the network. This configuration variable only has to be implemented when NetworkConnectionProfile is implemented.		

2.1.2. AllowNewSessionsPendingFirmwareUpdate

Required	no		
Component	componentName	ChargingStation	
Variable	variableName	AllowNewSessionsPendingFirmwareUpdate	
	variableAttributes	mutability	ReadWrite
	variableCharacteristics	dataType	boolean
Description	Indicates whether new sessions can be started on EVSEs, while Charging Station is waiting for all EVSEs to become Available in order to start a pending firmware update. When a firmware update is waiting to be installed and this variable exists and has the value <i>true</i> , then, the Charging Station will not set free EVSEs to Unavailable, pending the update. This means that it may take longer until there is a point in time when all EVSEs of the Charging Station are free and it can perform the firmware update.		

2.1.3. DefaultMessageTimeout

Required	yes		
Component	componentName	OCPPCommCtrlr	

Variable	variableName	MessageTimeout	
	variableInstance	Default	
	variableAttributes	mutability	ReadOnly
	variableCharacteristics	unit	seconds
		dataType	integer
Description	The purpose of the message timeout is to be able to consider a request message as not sent and continue with other tasks when the message did not arrive due to communication errors or software failure. The message timeout setting in a Charging Station can be configured in the messageTimeout field in the <i>NetworkConnectionProfile</i> .		

2.1.4. FileTransferProtocols

Required	yes		
Component	componentName	OCPPCommCtrlr	
Variable	variableName	FileTransferProtocols	
	variableAttributes	mutability	ReadOnly
	variableCharacteristics	dataType	MemberList
Description	List of supported file transfer protocols. Possible values: FTP, FTPS, HTTP, HTTPS, SFTP.		

2.1.5. HeartbeatInterval

Required	no		
Component	componentName	OCPPCommCtrlr	
Variable	variableName	HeartbeatInterval	
	variableAttributes	mutability	ReadWrite
	variableCharacteristics	unit	seconds
		dataType	integer
		minLimit	1
Description	Interval of inactivity (no OCPP exchanges) with CSMS after which the Charging Station should send HeartbeatRequest .		

2.1.6. NetworkConfigurationPriority

Required	yes		
Component	componentName	OCPPCommCtrlr	
Variable	variableName	NetworkConfigurationPriority	
	variableAttributes	attributeType	Actual
		mutability	ReadWrite
	variableCharacteristics	dataType	SequenceList
		valueList	List of possible values
Description	A comma separated ordered list of the priority of the possible Network Connection Profiles. The list of possible available profile slots for the network configuration profiles SHALL be reported, via the valueList characteristic of this Variable.		

2.1.7. NetworkProfileConnectionAttempts

Required	yes		
Component	componentName	OCPPCommCtrlr	

Variable	variableName	NetworkProfileConnectionAttempts	
	variableAttributes	mutability	ReadWrite
	variableCharacteristics	dataType	integer
Description	Specifies the number of connection attempts the Charging Station executes before switching to a different profile.		

2.1.8. OfflineThreshold

Required	yes		
Component	componentName	OCPPCommCtrlr	
Variable	variableName	OfflineThreshold	
	variableAttributes	mutability	ReadWrite
	variableCharacteristics	unit	seconds
		dataType	integer
Description	When the offline period of a Charging Station exceeds the OfflineThreshold it is recommended to send a StatusNotificationRequest for all its Connectors when the Charging Station is back online.		

2.1.9. QueueAllMessages

Required	no		
Component	componentName	OCPPCommCtrlr	
Variable	variableName	QueueAllMessages	
	variableAttributes	mutability	ReadWrite
	variableCharacteristics	dataType	boolean
		When this variable is set to <i>true</i> , the Charging Station will queue all message until they are delivered to the CSMS. When set to <i>false</i> the Charging Station will only queue Transaction related messages as required in: E04.FR.01. and other requirements When this variable is the to <i>true</i> , and the Charging Station is running low on memory, the Charging Station SHALL drop TransactionEvent messages last, and when dropping measurements/meter data, the Charging Station SHALL drop intermediate values first (1st value, 3th value, 5th etc), not start dropping values from the beginning or end of the measurements/meter data. Default = false	

2.1.10. MessageAttemptsTransactionEvent

Required	yes		
Component	componentName	OCPPCommCtrlr	
Variable	variableName	MessageAttempts	
	variableInstance	TransactionEvent	
	variableAttributes	mutability	ReadWrite
	variableCharacteristics	dataType	integer
Description	How often the Charging Station should try to submit a TransactionEventRequest message when the CSMS fails to process it.		

2.1.11. MessageAttemptIntervalTransactionEvent

Required	yes		
Component	componentName	OCPPCommCtrlr	
Variable	variableName	MessageAttemptInterval	
	variableInstance	TransactionEvent	
	variableAttributes	attributeType	Actual
		mutability	ReadWrite
	variableCharacteristics	unit	seconds
		dataType	integer

Description	How long the Charging Station should wait before resubmitting a TransactionEventRequest message that the CSMS failed to process.	
--------------------	--	--

2.1.12. UnlockOnEVSideDisconnect

Required	yes	
Component	componentName	OCPPCommCtrlr
Variable	variableName	UnlockOnEVSideDisconnect
	variableAttributes	mutability ReadWrite/ReadOnly
Description	variableCharacteristics	dataType boolean
	When set to true, the Charging Station SHALL unlock the cable on the Charging Station side when the cable is unplugged at the EV. For an EVSE with only fixed cables, the mutability SHALL be ReadOnly and the actual value SHALL be false. For a charging station with fixed cables and sockets, the variable is only applicable to the sockets.	

2.1.13. WebSocketPingInterval

Required	no	
Component	componentName	OCPPCommCtrlr
Variable	variableName	WebSocketPingInterval
	variableAttributes	mutability ReadWrite
Description	variableCharacteristics	unit seconds dataType integer
	Only relevant for websocket implementations. 0 disables client side websocket Ping/Pong. In this case there is either no ping/pong or the server initiates the ping and client responds with Pong. Positive values are interpreted as number of seconds between pings. Negative values are not allowed. SetConfiguration is expected to return a <i>Rejected</i> result.	

2.1.14. ResetRetries

Required	yes	
Component	componentName	OCPPCommCtrlr
Variable	variableName	ResetRetries
	variableAttributes	mutability ReadWrite
Description	Number of times to retry a reset of the Charging Station when a reset was unsuccessful.	

2.1.15. MessageFieldLength

Required	no	
Component	componentName	OCPPCommCtrlr
Variable	variableName	FieldLength
	variableInstance	<message>.<field>
Description	variableAttributes	mutability ReadOnly
	variableCharacteristics	dataType integer
This variable is used to report the length of <field> in <message> when it is larger than the length that is defined in the standard OCPP message schema.		

2.1.16. ItemsPerMessageGetReport

Required	yes	
Component	componentName	DeviceDataCtrlr

Variable	variableName	ItemsPerMessage	
	variableInstance	GetReport	
	variableAttributes	mutability	ReadOnly
	variableCharacteristics	dataType	integer
Description	Maximum number of ComponentVariable entries that can be sent in one getReportRequest or GetMonitoringReportRequest message.		

2.1.17. ItemsPerMessageGetVariables

Required	yes		
Component	componentName	DeviceDataCtrlr	
Variable	variableName	ItemsPerMessage	
	variableInstance	GetVariables	
	variableAttributes	mutability	ReadOnly
	variableCharacteristics	dataType	integer
Description	Maximum number of GetVariableData objects in GetVariablesRequest .		

2.1.18. BytesPerMessageGetReport

Required	yes		
Component	componentName	DeviceDataCtrlr	
Variable	variableName	BytesPerMessage	
	variableInstance	GetReport	
	variableAttributes	mutability	ReadOnly
	variableCharacteristics	dataType	integer
Description	Message Size (in bytes) - puts constraint on getReportRequest or GetMonitoringReportRequest message size.		

2.1.19. BytesPerMessageGetVariables

Required	yes		
Component	componentName	DeviceDataCtrlr	
Variable	variableName	BytesPerMessage	
	variableInstance	GetVariables	
	variableAttributes	mutability	ReadOnly
	variableCharacteristics	dataType	integer
Description	Message Size (in bytes) - puts constraint on GetVariablesRequest message size.		

2.1.20. ConfigurationValueSize

Required	no		
Component	componentName	DeviceDataCtrlr	
Variable	variableName	ConfigurationValueSize	
	variableAttributes	mutability	ReadOnly
	variableCharacteristics	dataType	integer
	maxLimit	1000	
Description	This Configuration Variable can be used to limit the following fields: SetVariableData.attributeValue and VariableCharacteristics.valueList. The max size of these values will always remain equal.		

2.1.21. ReportingValueSize

Required	no
-----------------	----

Component	componentName	DeviceDataCtrlr	
Variable	variableName	ReportingValueSize	
	variableAttributes	mutability	ReadOnly
	variableCharacteristics	dataType	integer
		maxLimit	2500
Description	This Configuration Variable can be used to limit the following fields: GetVariableResult.attributeValue, VariableAttribute.value and EventData.actualValue. The max size of these values will always remain equal.		

2.1.22. ItemsPerMessageSetVariables

Required	yes		
Component	componentName	DeviceDataCtrlr	
Variable	variableName	ItemsPerMessage	
	variableInstance	SetVariables	
	variableAttributes	mutability	ReadOnly
	variableCharacteristics	dataType	integer
Description	Maximum number of SetVariableData objects in SetVariablesRequest .		

2.1.23. BytesPerMessageSetVariables

Required	yes		
Component	componentName	DeviceDataCtrlr	
Variable	variableName	BytesPerMessage	
	variableInstance	SetVariables	
	variableAttributes	mutability	ReadOnly
	variableCharacteristics	dataType	integer
Description	Message Size (in bytes) - puts constraint on SetVariablesRequest message size.		

2.1.24. DateTime

Required	yes		
Component	componentName	ClockCtrlr	
Variable	variableName	DateTime	
	variableAttributes	mutability	ReadOnly
	variableCharacteristics	dataType	DateTime
	Description	Contains the current date and time.	

2.1.25. NtpSource

Required	no		
Component	componentName	ClockCtrlr	
Variable	variableName	NtpSource	
	variableAttributes	mutability	ReadWrite
	variableCharacteristics	dataType	OptionList
		valuesList	DHCP, manual
Description	When an NTP client is implemented, this variable can be used to configure the client: Use the NTP server provided via DHCP, or use the manually configured NTP server.		

2.1.26. NtpServerUri

Required	no
-----------------	----

Component	componentName	ClockCtrlr
Variable	variableName	NtpServerUri
	variableInstance	Single digit, multiple servers allowed, primary NtpServer has instance '1', the secondary has instance '2'. etc
	variableAttributes	mutability ReadWrite
	variableCharacteristics	dataType string
Description	When an NTP client is implemented, this variable can be used to configure the client: This contains the address of the NTP server. Multiple NTP servers can be configured. These can be back-up NTP servers. If the NTP client supports it, it can also connect to multiple NTP servers simultaneous to get a more reliable time source.	

2.1.27. TimeOffset

Required	no
Component	componentName
Variable	variableName
	variableAttributes
	variableCharacteristics
Description	Configured current local time offset in the format: "+01:00", "-02:00" etc. When a TimeOffset is used, it is advised not to implement: TimeZone . If a Charging Station has implemented both TimeOffset and TimeZone it is RECOMMENDED to not use both at the same time. The time offset is for display purposes.

2.1.28. NextTimeOffsetTransitionDateTime

Required	no
Component	componentName
Variable	variableName
	variableAttributes
	variableCharacteristics
Description	Date time of the next time offset transition. On this date time, the clock displayed to the EV driver will be given the new offset as configured via ' TimeOffsetNextTransition '. This can be used to manually configure the next start or end of a daylight saving time period.

2.1.29. TimeOffsetNextTransition

Required	no
Component	componentName
Variable	variableName
	variableInstance
	variableAttributes
	variableCharacteristics
Description	Next local time offset in the format: "+01:00", "-02:00" etc. New offset that will be set on the next time offset transition as configured via ' NextTimeOffsetTransitionDateTime '. This can be used to manually configure the offset for the start or end of the daylight saving time period.

2.1.30. TimeSource

Required	yes
-----------------	-----

Component	componentName	ClockCtrlr	
Variable	variableName	TimeSource	
	variableAttributes	mutability	ReadWrite
	variableCharacteristics	dataType	SequenceList
		valuesList	List of all implemented time sources. Possible values: Heartbeat, NTP, GPS, RealTimeClock, MobileNetwork, RadioTimeTransmitter
Description	<p>Via this variable, the Charging Station provides the CSMS with the option to configure a clock source, if more than 1 are implemented.</p> <p>By providing a list of possible sources, the CSO can configure fallback sources.</p> <p>Example: "NTP,Heartbeat" means, use NTP, but when none of the NTP servers responses, use time synchronization via Heartbeat.</p> <p>NOTE: RadioTimeTransmitter: At various locations around the globe, low-frequency radio transmitters provide accurate local time information e.g. DCF77 in Germany, MSF in the United Kingdom, JJY in Japan etc. Such a radio time clock can be used as a time source for a Charging Station. The Charging Station shall convert the broadcasted time to UTC. For this TimeZone, TimeOffset, 'NextTimeOffsetTransitionDateTime' and 'TimeOffsetNextTransition' can be used.</p>		

2.1.31. TimeZone

Required	no		
Component	componentName	ClockCtrlr	
Variable	variableName	TimeZone	
	variableAttributes	mutability	ReadWrite
	variableCharacteristics	dataType	string
Description	<p>Configured current local time zone in the format: "Europe/Oslo", "Asia/Singapore" etc.</p> <p>When a time zone is used, it is advised not to implement: TimeOffset. If a Charging Station has implemented both TimeOffset and TimeZone it is RECOMMENDED to not use both at the same time.</p> <p>The time zone is for display purposes.</p>		

2.1.32. TimeAdjustmentReportingThreshold

Required	no		
Component	componentName	ClockCtrlr	
Variable	variableName	TimeAdjustmentReportingThreshold	
	variableAttributes	mutability	ReadWrite
	variableCharacteristics	dataType	integer
Description	When the clock time is adjusted forwards or backwards for more than TimeAdjustmentReportingThreshold number of seconds, a SecurityEventNotification("SettingSystemTime") is sent by the charging station. A reasonable value is 20 seconds.		

2.1.33. CustomImplementationEnabled

Required	no		
Component	componentName	CustomizationCtrlr	
Variable	variableName	CustomImplementationEnabled	
	variableInstance	<VendorId>	
	variableAttributes	mutability	ReadWrite
	variableCharacteristics	dataType	boolean

Description	This standard configuration variable can be used to enable/disable custom implementations that the Charging Station supports. It is recommended to first check if the custom behavior is able to be implemented using the device model, otherwise DataTransfer message(s) and/or CustomData fields can be used.
--------------------	--

2.2. Security related

2.2.1. BasicAuthPassword

The basic authentication password is used for HTTP Basic Authentication. The configuration value is write-only, so that it cannot be accidentally stored in plaintext by the CSMS when it reads out all configuration values.

Required	no			
Component	componentName	SecurityCtrlr		
Variable	variableName	BasicAuthPassword		
	variableAttributes	mutability	WriteOnly	
	variableCharacteristics	dataType	passwordString	
		maxLimit	40 (Max length of the BasicAuthPassword)	
Description	The basic authentication password is used for HTTP Basic Authentication. The password SHALL be a randomly chosen passwordString with a sufficiently high entropy, consisting of minimum 16 and maximum 40 characters (alpha-numeric characters and the special characters allowed by passwordString). The password SHALL be sent as a UTF-8 encoded string (NOT encoded into octet string or base64). This configuration variable is write-only, so that it cannot be accidentally stored in plaintext by the CSMS when it reads out all configuration variables. This configuration variable is required unless only "security profile 3 - TLS with client side certificates" is implemented.			

2.2.2. Identity

Required	no			
Component	componentName	SecurityCtrlr		
Variable	variableName	Identity		
	variableAttributes	mutability	ReadOnly or ReadWrite	
	variableCharacteristics	dataType	identifierString	
		maxLimit	48 (Charging Station Identity)	
Description	The Charging Station identity. identity is an identifierString, however because this value is also used as the basic authentication username, the colon character ':' SHALL not be used. Maximum length was chosen to ensure compatibility with EVSE ID from [EMI3-BO] "Part 2: business objects".			

2.2.3. OrganizationName

Required	yes			
Component	componentName	SecurityCtrlr		
Variable	variableName	OrganizationName		
	variableAttributes	mutability	ReadWrite	
	variableCharacteristics	dataType	string	
		This configuration variable is used to set the organization name of the CSO or an organization trusted by the CSO. It is used to set the O (organizationName) RDN in the subject field of the client certificate. See also A00.FR.509.		

2.2.4. CertificateEntries

Required	yes			
Component	componentName	SecurityCtrlr		

Variable	variableName	CertificateEntries	
	variableAttributes	mutability	ReadOnly
	variableCharacteristics	dataType	integer
		maxLimit	Maximum number of Certificates installed at any time.
Description	Amount of Certificates currently installed on the Charging Station.		

2.2.5. SecurityProfile

Required	yes		
Component	componentName	SecurityCtrlr	
Variable	variableName	SecurityProfile	
	variableAttributes	mutability	ReadOnly
	variableCharacteristics	dataType	integer
Description	This configuration variable is used to report the security profile used by the Charging Station.		

2.2.6. AdditionalRootCertificateCheck

Required	no		
Component	componentName	SecurityCtrlr	
Variable	variableName	AdditionalRootCertificateCheck	
	variableAttributes	mutability	ReadOnly
	variableCharacteristics	dataType	boolean
Description	<p>When set to true, only one certificate (plus a temporarily fallback certificate) of certificateType CSMSRootCertificate is allowed to be installed at a time. When installing a new CSMS Root certificate, the new certificate SHALL replace the old one AND the new CSMS Root Certificate MUST be signed by the old CSMS Root Certificate it is replacing.</p> <p>This configuration variable is required unless only "security profile 1 - Unsecured Transport with Basic Authentication" is implemented. Please note that security profile 1 SHOULD only be used in trusted networks.</p> <p><i>Note: When using this additional security mechanism please be aware that the Charging Station needs to perform a full certificate chain verification when the new CSMS Root certificate is being installed. However, once the old CSMS Root certificate is set as the fallback certificate, the Charging Station needs to perform a partial certificate chain verification when verifying the server certificate during the TLS handshake. Otherwise the verification will fail once the old CSMS Root (fallback) certificate is either expired or removed.</i></p> <p><i>Note 2: The statement that the variable is required, means that the configuration variable must be present, but does NOT indicate that the feature must be implemented. This is an optional feature. By setting the value to false, the Charging Station indicates that it does not support this feature, whereas true means that it does support the feature.</i></p>		

2.2.7. MaxCertificateChainSize

Required	no		
Component	componentName	SecurityCtrlr	
Variable	variableName	MaxCertificateChainSize	
	variableAttributes	mutability	ReadOnly
	variableCharacteristics	dataType	integer
		maxLimit	10000
Description	This configuration variable can be used to limit the size of the 'certificateChain' field from the CertificateSignedRequest PDU. This value SHOULD NOT be set too small. The smaller this value, the less security architectures the Charging Station will support. It is RECOMMENDED to set at least a size of 5600. This will allow the Charging Station to support most security architectures.		

2.2.8. CertSigningWaitMinimum

Required	no			
Component	componentName	SecurityCtrlr		
Variable	variableName	CertSigningWaitMinimum		
	variableAttributes	mutability	ReadWrite	
	variableCharacteristics	unit	seconds	
		dataType	integer	
Description	This configuration variable defines how long the Charging Station has to wait before generating another CSR, in the case the CSMS accepts the SignCertificateRequest, but never returns the signed certificate back. This value will be doubled after every attempt. The amount of attempts is configured at CertSigningRepeatTimes . If the certificate signing process is slow, this setting allows the CSMS to tell the Charging Station to allow more time.			

2.2.9. CertSigningRepeatTimes

Required	no			
Component	componentName	SecurityCtrlr		
Variable	variableName	CertSigningRepeatTimes		
	variableAttributes	mutability	ReadWrite	
	variableCharacteristics	dataType	integer	
		This variable can be used to configure the amount of times the Charging Station SHALL double the previous back-off time, starting with the number of seconds configured at CertSigningWaitMinimum , every time the back-off time expires without having received the CertificateSignedRequest containing the from the CSR generated signed certificate. When the maximum number of increments is reached, the Charging Station SHALL stop resending the SignCertificateRequest, until it is requested by the CSMS using a TriggerMessageRequest.		

2.3. Authorization related

2.3.1. AuthEnabled

Required	no			
Component	componentName	AuthCtrlr		
Variable	variableName	Enabled		
	variableAttributes	mutability	ReadWrite	
	variableCharacteristics	dataType	boolean	
		If set to <code>false</code> , then no authorization is done before starting a transaction or when reading an idToken. If an idToken was provided, then it will be put in the <code>idToken</code> field of the TransactionEventRequest. If no idToken was provided, then <code>idToken</code> in TransactionEventRequest will be left empty and type is set to <code>NoAuthorization</code> .		

2.3.2. AdditionalInfoItemsPerMessage

Required	no			
Component	componentName	AuthCtrlr		
Variable	variableName	AdditionalInfoItemsPerMessage		
	variableAttributes	mutability	ReadOnly	
	variableCharacteristics	dataType	integer	
		Maximum number of AdditionalInfo items that can be sent in one message. This configuration variable only has to be implemented when AdditionalInfo is implemented.		

2.3.3. OfflineTxForUnknownIdEnabled

Required	no			
Component	componentName	AuthCtrlr		

Variable	variableName	OfflineTxForUnknownIdEnabled	
	variableAttributes	mutability	ReadWrite
	variableCharacteristics	dataType	boolean
Description	If this key exists, the Charging Station supports Unknown Offline Authorization . If this key reports a value of <i>true</i> , Unknown Offline Authorization is enabled.		

2.3.4. AuthorizeRemoteStart

Required	yes		
Component	componentName	AuthCtrlr	
Variable	variableName	AuthorizeRemoteStart	
	variableAttributes	mutability	ReadOnly or ReadWrite. Choice is up to Charging Station implementation.
	variableCharacteristics	dataType	boolean
Description	Whether a remote request to start a transaction in the form of RequestStartTransactionRequest message should be authorized beforehand like a local action to start a transaction.		

2.3.5. LocalAuthorizeOffline

Required	yes		
Component	componentName	AuthCtrlr	
Variable	variableName	LocalAuthorizeOffline	
	variableAttributes	mutability	ReadWrite
	variableCharacteristics	dataType	boolean
Description	Whether the Charging Station, when <i>Offline</i> , will start a transaction for locally-authorized identifiers.		

2.3.6. LocalPreAuthorize

Required	yes		
Component	componentName	AuthCtrlr	
Variable	variableName	LocalPreAuthorize	
	variableAttributes	mutability	ReadWrite
	variableCharacteristics	dataType	boolean
Description	Whether the Charging Station, when online, will start a transaction for locally-authorized identifiers without waiting for or requesting an AuthorizeResponse from the CSMS.		

2.3.7. MasterPassGroupId

Required	no		
Component	componentName	AuthCtrlr	
Variable	variableName	MasterPassGroupId	
	variableAttributes	mutability	ReadWrite
	variableCharacteristics	dataType	string
		maxLimit	36 (The maximum string length of MasterPassGroupId)
Description	IdTokens that have this id as groupId belong to the Master Pass Group. Meaning they can stop any ongoing transaction, but cannot start transactions. This can, for example, be used by law enforcement personal to stop any ongoing transaction when an EV has to be towed away.		

2.3.8. DisableRemoteAuthorization

Required	no
-----------------	----

Component	componentName	AuthCtrlr	
Variable	variableName	DisableRemoteAuthorization	
	variableAttributes	mutability	ReadWrite
	variableCharacteristics	dataType	boolean
Description	When set to <i>true</i> this instructs the Charging Station to not issue any AuthorizationRequests, but only use Authorization Cache and Local Authorization List to determine validity of idTokens. <i>Note: The difference with DisablePostAuthorize is that this variable disables all authorization with CSMS, whereas DisablePostAuthorize only disables re-authorization of tokens that are as not-Accepted in the Authorization Cache or Local Authorization List.</i>		

2.4. Authorization Cache related

2.4.1. AuthCacheEnabled

NOTE When the value of this variable is changed, the content of the authorization cache should not be altered.

Required	no		
Component	componentName	AuthCacheCtrlr	
Variable	variableName	Enabled	
	variableAttributes	mutability	ReadWrite
	variableCharacteristics	dataType	boolean
Description	If this variable exists and reports a value of <i>true</i> , Authorization Cache is enabled.		

2.4.2. AuthCacheAvailable

Required	no		
Component	componentName	AuthCacheCtrlr	
Variable	variableName	Available	
	variableAttributes	mutability	ReadOnly
	variableCharacteristics	dataType	boolean
Description	If this variable exists and reports a value of <i>true</i> , Authorization Cache is supported, but not necessarily enabled.		

2.4.3. AuthCacheLifeTime

Required	no		
Component	componentName	AuthCacheCtrlr	
Variable	variableName	LifeTime	
	variableAttributes	mutability	ReadWrite
	variableCharacteristics	unit	Seconds
		dataType	integer
Description	Indicates how long it takes until a token expires in the authorization cache since it is last used.		

2.4.4. AuthCacheStorage

Required	no		
Component	componentName	AuthCacheCtrlr	
Variable	variableName	Storage	
	variableAttributes	mutability	ReadOnly
	variableCharacteristics	dataType	integer
		maxLimit	The maximum number of bytes

Description	Indicates the number of bytes currently used by the Authorization Cache . MaxLimit indicates the maximum number of bytes that can be used by the Authorization Cache .	
--------------------	--	--

2.4.5. AuthCachePolicy

Required	no	
Component	componentName	AuthCacheCtrlr
Variable	variableName	Policy
	variableAttributes	mutability ReadWrite
	variableCharacteristics	dataType OptionList
		valuesList LRU, LFU, FIFO, CUSTOM
Description	Cache Entry Replacement Policy: least recently used, least frequently used, first in first out, other custom mechanism.	

2.4.6. AuthCacheDisablePostAuthorize

Required	no	
Component	componentName	AuthCacheCtrlr
Variable	variableName	DisablePostAuthorize
	variableAttributes	mutability ReadWrite
	variableCharacteristics	dataType boolean
Description	When set to <i>true</i> this variable disables the behavior to request authorization for an idToken that is stored in the cache with a status other than Accepted, as stated in C10.FR.03 and C12.FR.05.	

2.5. Local Authorization List Management related

2.5.1. LocalAuthListEnabled

Required	no	
Component	componentName	LocalAuthListCtrlr
Variable	variableName	Enabled
	variableAttributes	mutability ReadWrite
	variableCharacteristics	dataType boolean
Description	If this variable exists and reports a value of <i>true</i> , Local Authorization List is enabled.	

2.5.2. LocalAuthListEntries

Required	when LocalAuthListAvailable is <i>true</i>	
Component	componentName	LocalAuthListCtrlr
Variable	variableName	Entries
	variableAttributes	mutability ReadOnly
	variableCharacteristics	dataType integer
		maxLimit The maximum number of IdTokens that can be stored in the Local Authorization List .
Description	Amount of IdTokens currently in the Local Authorization List . The maxLimit of this variable SHALL be provided to report the maximum number of IdTokens that can be stored in the Local Authorization List .	

2.5.3. LocalAuthListAvailable

Required	no	
Component	componentName	LocalAuthListCtrlr

Variable	variableName	Available	
	variableAttributes	mutability	ReadOnly
	variableCharacteristics	dataType	boolean
Description	If this variable exists and reports a value of <i>true</i> , Local Authorization List is supported.		

2.5.4. ItemsPerMessageSendLocalList

Required	when LocalAuthListAvailable is <i>true</i>		
Component	componentName	LocalAuthListCtrlr	
Variable	variableName	ItemsPerMessage	
	variableAttributes	mutability	ReadOnly
	variableCharacteristics	dataType	integer

2.5.5. BytesPerMessageSendLocalList

Required	when LocalAuthListAvailable is <i>true</i>		
Component	componentName	LocalAuthListCtrlr	
Variable	variableName	BytesPerMessage	
	variableAttributes	mutability	ReadOnly
	variableCharacteristics	dataType	integer

2.5.6. LocalAuthListStorage

Required	no		
Component	componentName	LocalAuthListCtrlr	
Variable	variableName	Storage	
	variableAttributes	mutability	ReadOnly
	variableCharacteristics	dataType	integer
		maxLimit	The maximum number of bytes
Description	Indicates the number of bytes currently used by the Local Authorization List . MaxLimit indicates the maximum number of bytes that can be used by the Local Authorization List .		

2.5.7. LocalAuthListDisablePostAuthorize

Required	no		
Component	componentName	LocalAuthListCtrlr	
Variable	variableName	DisablePostAuthorize	
	variableAttributes	mutability	ReadWrite
	variableCharacteristics	dataType	boolean
Description	When set to <i>true</i> this variable disables the behavior to request authorization for an idToken that is stored in the local authorization list with a status other than Accepted, as stated in C14.FR.03.		

2.6. Transaction related

2.6.1. EVConnectionTimeOut

Required	yes		
Component	componentName	TxCtrlr	

Variable	variableName	EVConnectionTimeOut	
	variableAttributes	mutability	ReadWrite
	variableCharacteristics	unit	seconds
		dataType	integer
Description	Interval from between "starting" of a transaction until incipient transaction is automatically canceled, due to failure of EV driver to (correctly) insert the charging cable connector(s) into the appropriate socket(s). The Charging Station SHALL go back to the original state, probably: 'Available'. "Starting" might be the swiping of the RFID, pressing a start button, a RequestStartTransactionRequest being received etc.		

2.6.2. StopTxOnEVSideDisconnect

Required	yes		
Component	componentName	TxCtrlr	
Variable	variableName	StopTxOnEVSideDisconnect	
	variableAttributes	mutability	ReadWrite or ReadOnly, depending on Charging Station implementation.
	variableCharacteristics	dataType	boolean
Description	When set to <i>true</i> , the Charging Station SHALL deauthorize the transaction when the cable is unplugged from the EV.		

2.6.3. TxBeforeAcceptedEnabled

Required	no		
Component	componentName	TxCtrlr	
Variable	variableName	TxBeforeAcceptedEnabled	
	variableAttributes	mutability	ReadWrite
	variableCharacteristics	dataType	boolean
Description	With this configuration variable the Charging Station can be configured to allow charging before having received a BootNotificationResponse with RegistrationStatus : Accepted. See: Transactions before being accepted by a CSMS .		

2.6.4. TxStartPoint

Required	yes		
Component	componentName	TxCtrlr	
Variable	variableName	TxStartPoint	
	variableAttributes	mutability	ReadOnly or ReadWrite. Choice is up to Charging Station implementation.
	variableCharacteristics	dataType	MemberList
		valueList	See TxStartStopPoint values for allowed values. It is not required to implement all possible values.
Description	Defines when the Charging Station starts a new transaction: first transactioneventRequest : eventType = Started. When any event in the given list occurs, the Charging Station SHALL start a transaction. The Charging Station SHALL only send the Started event once for every transaction. It is advised to put all events that should be part of a transaction in the list, in case the start event never occurs. Because the possible events don't always have to come in the same order it is possible to provide a list of events. Which ever comes first will then cause a transaction to be started. For example: EVConnected, Authorized would mean that a transaction is started when an EV is detected (Cable is connected), or when an EV Driver swipes his RFID card en the CSMS successfully authorizes the ID for charging.		

2.6.5. TxStopPoint

Required	yes		
Component	componentName	TxCtrlr	

Variable	variableName	TxStopPoint	
	variableAttributes	mutability	ReadOnly or ReadWrite. Choice is up to Charging Station implementation.
	variableCharacteristics	dataType	MemberList
		valueList	See TxStartStopPoint values for allowed values. It is not required to implement all possible values.
Description	Defines when the Charging Station ends a transaction: last transactioneventRequest : eventType = Ended. When any event in the given list is no longer valid, the Charging Station SHALL end the transaction. The Charging Station SHALL only send the Ended event once for every transaction.		

2.6.6. TxStartStopPoint values

2.6.6.1. TxStartPoint values

The following table lists the values allowed for the [TxStartPoint](#) variable. These values represent logical steps or events that (may) occur during a charging session. When such an event occurs, and it is listed in in the [TxStartPoint](#) variable, then this marks the start of a transaction.

Value	Description
ParkingBayOccupancy	An object (probably an EV) is detected in the parking/charging bay.
EVConnected	Both ends of the Charging Cable have been connected (if this can be detected, else detection of a cable being plugged into the socket), or for wireless charging: initial communication between EVSE and EV is established.
Authorized	Driver or EV has been authorized, this can also be some form of anonymous authorization like a start button.
PowerPathClosed	All preconditions for charging have been met, power can flow. This event is the logical AND of EVConnected and Authorized and should be used if a transaction is supposed to start when EV is connected and authorized. Despite its name, this event is not related to the state of the power relay. Note: There may be situations where PowerPathClosed does not imply that charging starts at that moment, e.g. because of delayed charging or a battery that is too hot.
EnergyTransfer	Energy is being transferred between EV and EVSE.
DataSigned	The moment when the signed meter value is received from the fiscal meter, that is used in the TransactionEventRequest with <i>context</i> = <code>Transaction.Begin</code> and <i>triggerReason</i> = <code>SignedDataReceived</code> . This TxStartPoint might be applicable when legislation exists that only allows a billable transaction to start when the first signed meter value has been received.

2.6.6.2. TxStopPoint values

The following table lists the values allowed for the [TxStopPoint](#) variable. These values represent logical steps or events that (may) occur during a charging session. When such an event occurs, and it is listed in in the [TxStopPoint](#) variable, then this marks the end of a transaction.

The values are the same as for [TxStartPoint](#), but in this case the meaning is different, since it refers to the ending of the event, rather than the start. For use with [TxStopPoint](#) each value should be interpreted as if it had "Not" prefixed to it. See the following table:

Value	Description
ParkingBayOccupancy	An object (probably an EV) is no longer detected in the parking/charging bay.
EVConnected	One or both ends of the Charging Cable have been disconnected (if this can be detected, else detection of a cable being unplugged from the socket), or for wireless charging: communication between EVSE and EV is lost .

Value	Description
Authorized	Driver or EV is no longer authorized, this can also be some form of anonymous authorization like a start button.
PowerPathClosed	All preconditions for charging are no longer met, power cannot flow. This event is the logical OR of EVConnected and Authorized and should be used if a transaction is supposed to end when EV is disconnected and/or deauthorized. It is exactly the same as having the values EVConnected, Authorized in TxStopPoint. Despite its name, this event is not related to the state of the power relay.
EnergyTransfer	Energy is not being transferred between EV and EVSE. This is not recommended to use as a TxStopPoint, because it will stop the transaction as soon as EV or EVSE (temporarily) suspend the charging.
DataSigned	This condition has no meaning as a TxStopPoint and should not be used as such.

2.6.7. MaxEnergyOnInvalidId

Required	no		
Component	componentName	TxCtrlr	
Variable	variableName	MaxEnergyOnInvalidId	
	variableAttributes	mutability	ReadWrite
	variableCharacteristics	unit	Wh
		dataType	integer
Description	Maximum amount of energy in Wh delivered when an identifier is deauthorized by the CSMS after start of a transaction.		

2.6.8. StopTxOnInvalidId

Required	yes		
Component	componentName	TxCtrlr	
Variable	variableName	StopTxOnInvalidId	
	variableAttributes	mutability	ReadWrite
	variableCharacteristics	dataType	boolean
Description	whether the Charging Station will deauthorize an ongoing transaction when it receives a non-Accepted authorization status in TransactionEventResponse for this transaction.		

2.7. Metering related

2.7.1. SampledDataEnabled

Required	no		
Component	componentName	SampledDataCtrlr	
Variable	variableName	Enabled	
	variableAttributes	mutability	ReadWrite
	variableCharacteristics	dataType	boolean
Description	If this variable reports a value of true, Sampled Data is enabled.		

2.7.2. SampledDataAvailable

Required	no		
Component	componentName	SampledDataCtrlr	

Variable	variableName	Available	
	variableAttributes	mutability	ReadOnly
	variableCharacteristics	dataType	boolean
Description	If this variable reports a value of <i>true</i> , Sampled Data is supported.		

2.7.3. SampledDataSignReadings

Required	no		
Component	componentName	SampledDataCtrlr	
Variable	variableName	SignReadings	
	variableAttributes	mutability	ReadWrite
	variableCharacteristics	dataType	boolean
Description	If set to <i>true</i> , the Charging Station SHALL include signed meter values in the TransactionEventRequest to the CSMS. Some Charging Stations might only be able to sign Transaction.Begin and Transaction.End meter values. When a Charging Station does not support signed meter values it SHALL NOT report this variable.		

2.7.4. SampledTxDxEndedMeasurands

Required	yes		
Component	componentName	SampledDataCtrlr	
Variable	variableName	TxEndedMeasurands	
	variableAttributes	mutability	ReadWrite
	variableCharacteristics	dataType	MemberList
		maxLimit	The maximum length of the CSV formatted string, to be defined by the implementer.
Description	Sampled measurands to be included in the <i>meterValues</i> element of TransactionEventRequest (<i>eventType = Ended</i>), every SampledTxDxEndedInterval seconds from the start of the transaction. The Charging Station reports the list of supported Measurands in VariableCharacteristicsType.valuesList of this variable. This way the CSMS knows which Measurands it can put in the TxEndedSampledData . When left empty, no sampled measurands SHALL be put into the TransactionEventRequest (<i>eventType = Ended</i>).		

2.7.5. SampledTxDxEndedInterval

Required	yes		
Component	componentName	SampledDataCtrlr	
Variable	variableName	TxEndedInterval	
	variableAttributes	mutability	ReadWrite
	variableCharacteristics	unit	seconds
		dataType	integer
Description	Interval between sampling of metering (or other) data, intended to be transmitted in the TransactionEventRequest (<i>eventType = Ended</i>) message. For transaction data (<i>evseld>0</i>), samples are acquired and transmitted only in the TransactionEventRequest (<i>eventType = Ended</i>) message. A value of "0" (numeric zero), by convention, is to be interpreted to mean that only the values taken at the <i>start</i> and <i>end</i> of a transaction should be transmitted (no intermediate values).		

2.7.6. SampledTxDxStartedMeasurands

Required	yes		
Component	componentName	SampledDataCtrlr	

Variable	variableName	TxStartedMeasurands	
	variableAttributes	mutability	ReadWrite
	variableCharacteristics	dataType	MemberList
		maxLimit	The maximum length of the CSV formated string, to be defined by the implementer.
Description	Sampled measurand(s) to be taken at the start of any transaction to be included in the meterValues field of the first TransactionEventRequest message send at the start of a transaction (eventType = Started). The Charging Station reports the list of supported Measurands in VariableCharacteristicsType.valuesList of this variable. This way the CSMS knows which Measurands it can put in the SampledDataTxStartedMeasurands . If the Charging Station has a meter, recommended to use as default: "Energy.Active.Import.Register"		

2.7.7. SampledDataTxUpdatedMeasurands

Required	yes		
Component	componentName	SampledDataCtrlr	
Variable	variableName	TxUpdatedMeasurands	
	variableAttributes	mutability	ReadWrite
	variableCharacteristics	dataType	MemberList
		maxLimit	The maximum length of the CSV formated string, to be defined by the implementer.
Description	Sampled measurands to be included in the <i>meterValues</i> element of every TransactionEventRequest (<i>eventType = Updated</i>), every SampledDataTxUpdatedInterval seconds from the start of the transaction. The Charging Station reports the list of supported Measurands in VariableCharacteristicsType.valuesList of this variable. This way the CSMS knows which Measurands it can put in the SampledDataTxUpdatedMeasurands . If the Charging Station has a meter, recommended to use as default: "Energy.Active.Import.Register"		

2.7.8. SampledDataTxUpdatedInterval

Required	yes		
Component	component Name	SampledDataCtrlr	
Variable	variableName	TxUpdatedInterval	
	variableAttributes	mutability	ReadWrite
	variableCharacteristics	unit	seconds
		dataType	integer
Description	Interval between sampling of metering (or other) data, intended to be transmitted via TransactionEventRequest (<i>eventType = Updated</i>) messages. For transaction data (evseld>0), samples are acquired and transmitted periodically at this interval from the start of the charging transaction. A value of "0" (numeric zero), by convention, is to be interpreted to mean that no sampled data should be transmitted during the transaction.		

2.7.9. AlignedDataEnabled

Required	no		
Component	componentName	AlignedDataCtrlr	
Variable	variableName	Enabled	
	variableAttributes	mutability	ReadWrite
	variableCharacteristics	dataType	boolean
Description	If this variable reports a value of <i>true</i> , Aligned Data is enabled.		

2.7.10. AlignedDataAvailable

Required	no		
Component	componentName	AlignedDataCtrlr	
Variable	variableName	Available	
	variableAttributes	mutability	ReadOnly
	variableCharacteristics	dataType	boolean
Description	If this variable reports a value of <i>true</i> , Aligned Data is supported.		

2.7.11. AlignedDataMeasurands

Required	yes		
Component	componentName	AlignedDataCtrlr	
Variable	variableName	Measurands	
	variableAttributes	mutability	ReadWrite
	variableCharacteristics	dataType	MemberList
Description		maxLimit	The maximum length of the CSV formatted string, to be defined by the implementer.
Clock-aligned measurand(s) to be included in MeterValuesRequest , every AlignedDataInterval seconds. For all the allowed values see: Measurand . The Charging Station reports the list of supported Measurands in VariableCharacteristicsType.valuesList of this variable. This way the CSMS knows which Measurands it can put in the AlignedDataMeasurands .			

2.7.12. AlignedDataInterval

Required	yes		
Component	componentName	AlignedDataCtrlr	
Variable	variableName	Interval	
	variableAttributes	mutability	ReadWrite
	variableCharacteristics	unit	seconds
Description		dataType	integer
Size (in seconds) of the clock-aligned data interval, intended to be transmitted in the MeterValuesRequest message. This is the size (in seconds) of the set of evenly spaced aggregation intervals per day, starting at 00:00:00 (midnight). For example, a value of 900 (15 minutes) indicates that every day should be broken into 96 15-minute intervals. When clock aligned data is being transmitted, the interval in question is identified by the start time and (optional) duration interval value, represented according to the ISO8601 standard. All "per-period" data (e.g. energy readings) should be accumulated (for "flow" type measurands such as energy), or averaged (for other values) across the entire interval (or partial interval, at the beginning or end of a transaction), and transmitted (if so enabled) at the end of each interval, bearing the interval start time timestamp. A value of "0" (numeric zero), by convention, is to be interpreted to mean that no clock-aligned data should be transmitted.			

2.7.13. AlignedDataSendDuringIdle

Required	no		
Component	componentName	AlignedDataCtrlr	
	evse	*	
Variable	variableName	SendDuringIdle	
	variableAttributes	mutability	ReadWrite
	variableCharacteristics	dataType	boolean
Description	If set to <i>true</i> , the Charging Station SHALL NOT send clock aligned meter values when a transaction is ongoing. When an EVSE is specified, it SHALL stop sending the clock aligned meter values for this EVSE when it has an ongoing transaction. When no EVSE is specified, it SHALL stop sending the clock aligned meter values when any transaction is ongoing on this Charging Station.		

2.7.14. AlignedDataSignReadings

Required	no		
Component	componentName	AlignedDataCtrlr	
Variable	variableName	SignReadings	
	variableAttributes	mutability	ReadWrite
Description	variableCharacteristics		
	dataType	boolean	
Description	If set to <i>true</i> , the Charging Station SHALL include signed meter values in the SampledValueType in the TransactionEventRequest to the CSMS for those measurands defined in AlignedDataTxEndedMeasurands . When a Charging Station does not support signed meter values it SHALL NOT report this variable.		

2.7.15. AlignedDataTxEndedMeasurands

Required	yes		
Component	componentName	AlignedDataCtrlr	
Variable	variableName	TxEndedMeasurands	
	variableAttributes	mutability	ReadWrite
Description	variableCharacteristics	dataType	MemberList
		maxLimit	The maximum length of the CSV formatted string, to be defined by the implementer.
Description	Clock-aligned periodic measurand(s) to be included in the <i>meterValues</i> element of TransactionEventRequest (<i>eventType</i> = <i>Ended</i>) for every AlignedDataTxEndedInterval of the transaction. The Charging Station reports the list of supported Measurands in VariableCharacteristicsType.valuesList of this variable. This way the CSMS knows which Measurands it can put in the TxEndedAlignedData . When left empty, no Clock-aligned measurands SHALL be put into the TransactionEventRequest (<i>eventType</i> = <i>Ended</i>).		

2.7.16. AlignedDataTxEndedInterval

Required	yes		
Component	componentName	AlignedDataCtrlr	
Variable	variableName	TxEndedInterval	
	variableAttributes	mutability	ReadWrite
Description	variableCharacteristics	unit	seconds
		dataType	integer
Description	Size (in seconds) of the clock-aligned data interval, intended to be transmitted in the TransactionEventRequest (<i>eventType</i> = <i>Ended</i>) message. This is the size (in seconds) of the set of evenly spaced aggregation intervals per day, starting at 00:00:00 (midnight). For example, a value of 900 (15 minutes) indicates that every day should be broken into 96 15-minute intervals. When clock aligned data is being collected, the interval in question is identified by the start time and (optional) duration interval value, represented according to the ISO8601 standard. All "per-period" data (e.g. energy readings) should be accumulated (for "flow" type measurands such as energy), or averaged (for other values) across the entire interval (or partial interval, at the beginning or end of a transaction), and transmitted (if so enabled) at the end of the transaction in 1 TransactionEventRequest (<i>eventType</i> = <i>Ended</i>) message.		

2.7.17. PublicKeyWithSignedMeterValue

Required	no		
Component	componentName	OCPPCommCtrlr	
Variable	variableName	PublicKeyWithSignedMeterValue	
	variableAttributes	mutability	ReadWrite
Description	variableCharacteristics	dataType	OptionList
		valueList	Never,OncePerTransaction,EveryMeterValue

Description	This Configuration Variable can be used to configure whether a public key needs to be sent with a signed meter value. Note, that the field is required, so it needs to be present as an empty string when the public key is not sent.	
--------------------	---	--

2.7.18. SampledDataRegisterValuesWithoutPhases

Required	no	
Component	componentName	SampledDataCtrlr
Variable	variableName	RegisterValuesWithoutPhases
	variableAttributes	mutability ReadWrite
	variableCharacteristics	dataType boolean
Description	If this variable reports a value of <i>true</i> , then meter values of measurand Energy.Active.Import.Register will only report the total energy over all phases without reporting the individual phase values. If this variable is absent or <i>false</i> , then the value for each phase is reported, possibly also with a total value (depending on the meter).	

2.8. Reservation related

2.8.1. ReservationEnabled

Required	no	
Component	componentName	ReservationCtrlr
Variable	variableName	Enabled
	variableAttributes	mutability ReadWrite
	variableCharacteristics	dataType boolean
Description	Whether Reservation is enabled.	

2.8.2. ReservationAvailable

Required	no	
Component	componentName	ReservationCtrlr
Variable	variableName	Available
	variableAttributes	mutability ReadOnly
	variableCharacteristics	dataType boolean
Description	Whether Reservation is supported.	

2.8.3. ReservationNonEvseSpecific

Required	no	
Component	componentName	ReservationCtrlr
Variable	variableName	NonEvseSpecific
	variableAttributes	mutability ReadOnly
	variableCharacteristics	dataType boolean
Description	If this configuration variable is present and set to <i>true</i> : Charging Station supports Reservation where EVSE id is not specified.	

2.9. Smart Charging related

2.9.1. SmartChargingEnabled

Required	no	
Component	componentName	SmartChargingCtrlr

Variable	variableName	Enabled	
	variableAttributes	mutability	ReadWrite
	variableCharacteristics	dataType	boolean
Description	Whether Smart Charging is enabled.		

2.9.2. SmartChargingAvailable

Required	no		
Component	componentName	SmartChargingCtrlr	
Variable	variableName	Available	
	variableAttributes	mutability	ReadOnly
	variableCharacteristics	dataType	boolean
Description	Whether Smart Charging is supported.		

2.9.3. ACPhaseSwitchingSupported

Required	no		
Component	componentName	SmartChargingCtrlr	
Variable	variableName	ACPhaseSwitchingSupported	
	variableAttributes	mutability	ReadOnly
	variableCharacteristics	dataType	boolean
Description	This variable can be used to indicate an on-load/in-transaction capability. If defined and true, this EVSE supports the selection of which phase to use for 1 phase AC charging.		

2.9.4. ChargingProfileMaxStackLevel

Required	yes		
Component	componentName	SmartChargingCtrlr	
Variable	variableName	ProfileStackLevel	
	variableAttributes	mutability	ReadOnly
	variableCharacteristics	dataType	integer
Description	Maximum acceptable value for <i>stackLevel</i> in a <i>ChargingProfile</i> . Since the lowest <i>stackLevel</i> is 0, this means that if SmartChargingCtrlr.ProfileStackLevel = 1, there can be at most 2 valid charging profiles per Charging Profile Purpose per EVSE.		

2.9.5. ChargingScheduleChargingRateUnit

Required	yes		
Component	componentName	SmartChargingCtrlr	
Variable	variableName	RateUnit	
	variableAttributes	mutability	ReadOnly
	variableCharacteristics	dataType	MemberList
Description	A list of supported quantities for use in a ChargingSchedule . Allowed values: 'A' and 'W'		

2.9.6. PeriodsPerSchedule

Required	yes		
Component	componentName	SmartChargingCtrlr	

Variable	variableName	PeriodsPerSchedule	
	variableAttributes	mutability	ReadOnly
	variableCharacteristics	dataType	integer
Description	Maximum number of periods that may be defined per ChargingSchedule .		

2.9.7. ExternalControlSignalsEnabled

Required	no		
Component	componentName	SmartChargingCtrlr	
Variable	variableName	ExternalControlSignalsEnabled	
	variableAttributes	mutability	ReadOnly or ReadWrite. Choice is up to Charging Station implementation.
	variableCharacteristics	dataType	boolean
Description	Indicates whether a Charging Station should respond to external control signals that influence charging.		

2.9.8. NotifyChargingLimitWithSchedules

Required	no		
Component	componentName	SmartChargingCtrlr	
Variable	variableName	NotifyChargingLimitWithSchedules	
	variableAttributes	mutability	ReadWrite
	variableCharacteristics	dataType	boolean
Description	Indicates if the Charging Station should include the externally set charging limit/schedule in the message when it sends a NotifyChargingLimitRequest message. This might increase the data usage significantly, especially when an external system sends new profiles/limits with a short interval. Default is false when omitted.		

2.9.9. Phases3to1

Required	no		
Component	componentName	SmartChargingCtrlr	
Variable	variableName	Phases3to1	
	variableAttributes	mutability	ReadOnly
	variableCharacteristics	dataType	boolean
Description	If defined and true, this Charging Station supports switching from 3 to 1 phase during a transaction.		

2.9.10. ChargingProfileEntries

Required	yes		
Component	componentName	SmartChargingCtrlr	
Variable	variableName	Entries	
	VariableInstance	ChargingProfiles	
	variableAttributes	mutability	ReadOnly
	variableCharacteristics	dataType	integer
		maxLimit	Maximum number of Charging profiles installed at any time.
Description	Amount of Charging profiles currently installed on the Charging Station.		

2.9.11. LimitChangeSignificance

Required	yes		
Component	componentName	SmartChargingCtrlr	

Variable	variableName	LimitChangeSignificance	
	variableAttributes	mutability	ReadWrite
	variableCharacteristics	dataType	decimal
Description	If at the Charging Station side a change in the limit in a ChargingProfile is lower than this percentage, the Charging Station MAY skip sending a NotifyChargingLimitRequest or a TransactionEventRequest message to the CSMS. It is RECOMMENDED to set this key to a low value. See Smart Charging signals to a Charging Station from multiple actors .		

2.10. Tariff & Cost related

2.10.1. TariffEnabled

Required	no		
Component	componentName	TariffCostCtrlr	
Variable	variableName	Enabled	
	variableInstance	Tariff	
	variableAttributes	mutability	ReadWrite
	variableCharacteristics	dataType	boolean
Description	Whether Tariff is enabled.		

2.10.2. TariffAvailable

Required	no		
Component	componentName	TariffCostCtrlr	
Variable	variableName	Available	
	variableInstance	Tariff	
	variableAttributes	mutability	ReadOnly
	variableCharacteristics	dataType	boolean
Description	Whether Tariff is supported.		

2.10.3. TariffFallbackMessage

Required for Charging Stations supporting Tariff Information.

Required	yes		
Component	componentName	TariffCostCtrlr	
Variable	variableName	TariffFallbackMessage	
	variableAttributes	mutability	ReadWrite
	variableCharacteristics	dataType	string
		maxLimit	255
Description	Message (and/or tariff information) to be shown to an EV Driver when there is no driver specific tariff information available.		

2.10.4. CostEnabled

Required	no		
Component	componentName	TariffCostCtrlr	
Variable	variableName	Enabled	
	variableInstance	Cost	
	variableAttributes	mutability	ReadWrite
	variableCharacteristics	dataType	boolean
Description	Whether Cost is enabled.		

2.10.5. CostAvailable

Required	no		
Component	componentName	TariffCostCtrlr	
Variable	variableName	Available	
	variableInstance	Cost	
	variableAttributes	mutability	ReadOnly
	variableCharacteristics	dataType	boolean
Description	Whether Cost is supported.		

2.10.6. TotalCostFallbackMessage

Required for Charging Stations supporting Tariff Information.

Required	yes		
Component	componentName	TariffCostCtrlr	
Variable	variableName	TotalCostFallbackMessage	
	variableAttributes	mutability	ReadWrite
	variableCharacteristics	dataType	string
		maxLimit	255
Description	Message to be shown to an EV Driver when the Charging Station cannot retrieve the cost for a transaction at the end of the transaction.		

2.10.7. Currency

Required for Charging Stations supporting Tariff Information.

Required	yes		
Component	componentName	TariffCostCtrlr	
Variable	variableName	Currency	
	variableAttributes	mutability	ReadWrite
	variableCharacteristics	dataType	string
		maxLimit	3
Description	Currency used by this Charging Station in a ISO 4217 [ISO4217] formatted currency code.		

2.11. Diagnostics related

2.11.1. MonitoringEnabled

Required	no		
Component	componentName	MonitoringCtrlr	
Variable	variableName	Enabled	
	variableAttributes	mutability	ReadWrite
	variableCharacteristics	dataType	boolean
		maxLimit	1
Description	Whether Monitoring is enabled.		

2.11.2. MonitoringAvailable

Required	no		
Component	componentName	MonitoringCtrlr	

Variable	variableName	Available	
	variableAttributes	mutability	ReadOnly
	variableCharacteristics	dataType	boolean
Description	Whether Monitoring is supported.		

2.11.3. ItemsPerMessageClearVariableMonitoring

Required	no		
Component	componentName	MonitoringCtrlr	
Variable	variableName	ItemsPerMessage	
	variableInstance	ClearVariableMonitoring	
	variableAttributes	mutability	ReadOnly
	variableCharacteristics	dataType	integer
Description	Maximum number of IDs in a ClearVariableMonitoringRequest .		

2.11.4. ItemsPerMessageSetVariableMonitoring

Required	yes		
Component	componentName	MonitoringCtrlr	
Variable	variableName	ItemsPerMessage	
	variableInstance	SetVariableMonitoring	
	variableAttributes	mutability	ReadOnly
	variableCharacteristics	dataType	integer
Description	Maximum number of setMonitoringData elements that can be sent in one setVariableMonitoringRequest message.		

2.11.5. BytesPerMessageClearVariableMonitoring

Required	no		
Component	componentName	MonitoringCtrlr	
Variable	variableName	BytesPerMessage	
	variableInstance	ClearVariableMonitoring	
	variableAttributes	mutability	ReadOnly
	variableCharacteristics	dataType	integer
Description	Message Size (in bytes) - puts constraint on ClearVariableMonitoringRequest message size.		

2.11.6. BytesPerMessageSetVariableMonitoring

Required	yes		
Component	componentName	MonitoringCtrlr	
Variable	variableName	BytesPerMessage	
	variableInstance	SetVariableMonitoring	
	variableAttributes	mutability	ReadOnly
	variableCharacteristics	dataType	integer
Description	Message Size (in bytes) - puts constraint on setVariableMonitoringRequest message size.		

2.11.7. OfflineMonitoringEventQueuingSeverity

Required	no		
Component	componentName	MonitoringCtrlr	

Variable	variableName	OfflineQueuingSeverity	
	variableAttributes	mutability	ReadWrite
	variableCharacteristics	dataType	integer
Description	When set and the Charging Station is <i>offline</i> , the Charging Station shall queue any notifyEventRequest messages triggered by a monitor with a severity number equal to or lower than the severity configured here. Value ranging from 0 (Emergency) to 9 (Debug).		

2.11.8. ActiveMonitoringBase

Required	no		
Component	componentName	MonitoringCtrlr	
Variable	variableName	ActiveMonitoringBase	
	variableAttributes	mutability	ReadOnly
	variableCharacteristics	dataType	OptionList
Description	Shows the currently used MonitoringBase. Valid values according MonitoringBaseEnumType: All, FactoryDefault, HardwiredOnly.		

2.11.9. ActiveMonitoringLevel

Required	no		
Component	componentName	MonitoringCtrlr	
Variable	variableName	ActiveMonitoringLevel	
	variableAttributes	mutability	ReadOnly
	variableCharacteristics	dataType	integer
Description	Shows the currently used MonitoringLevel. Valid values are severity levels of SetMonitoringLevelRequest: 0-9.		

2.12. Display Message related

2.12.1. DisplayMessageEnabled

Required	no		
Component	componentName	DisplayMessageCtrlr	
Variable	variableName	Enabled	
	variableAttributes	mutability	ReadWrite
	variableCharacteristics	dataType	boolean
Description	Whether Display Message is enabled.		

2.12.2. DisplayMessageAvailable

Required	no		
Component	componentName	DisplayMessageCtrlr	
Variable	variableName	Available	
	variableAttributes	mutability	ReadOnly
	variableCharacteristics	dataType	boolean
Description	Whether Display Message is supported.		

2.12.3. NumberOfDisplayMessages

Required	yes		
Component	componentName	DisplayMessageCtrlr	

Variable	variableName	DisplayMessages	
	variableAttributes	mutability	ReadOnly
	variableCharacteristics	dataType	integer
		maxLimit	Maximum number of different messages that can be configured in this Charging Station simultaneous, via SetDisplayMessageRequest .
Description	Amount of different messages that are currently configured in this Charging Station, via SetDisplayMessageRequest		

2.12.4. DisplayMessageSupportedFormats

Required	yes		
Component	componentName	DisplayMessageCtrlr	
Variable	variableName	SupportedFormats	
	variableAttributes	mutability	ReadOnly
	variableCharacteristics	dataType	MemberList
Description	List of message formats supported by this Charging Station. Possible values: MessageFormat .		

2.12.5. DisplayMessageSupportedPriorities

Required	yes		
Component	componentName	DisplayMessageCtrlr	
Variable	variableName	SupportedPriorities	
	variableAttributes	mutability	ReadOnly
	variableCharacteristics	dataType	MemberList
Description	List of the priorities supported by this Charging Station. Possible values: MessagePriority .		

2.13. Charging Infrastructure related

2.13.1. Available

Required	yes		
Components	componentName	ChargingStation	
		EVSE	
		Connector	
	evse	* (for EVSE and Connector)	
Variable	variableName	Available	
	variableAttributes	mutability	ReadOnly
	variableCharacteristics	dataType	boolean
Description	When <i>true</i> the Component exists and is locally configured/wired for use, but may not be (remotely) Enabled. This variable is required on any Component that can be reported by the Charging Station. As a minimum it shall exist on ChargingStation, EVSE and Connector.		
Note	<p>If any other variables are reported for a Component, then reporting <i>Available</i> does not add much value and may be omitted. However, the variable needs to exist, because it can be queried for by a GetCustomReport request for all Components that are 'available'.</p> <p>EVSE and Connector components are addressed on their respective tier. So, EVSE #1 is addressed as component EVSE on tier "evse = 1" and connector #1 on this EVSE is addressed as component Connector on tier "evse = 1, connector = 1".</p>		

2.13.2. AvailabilityState

Required	yes		
Components	componentName	ChargingStation	
		EVSE	
	evse	* (for EVSE)	
Variable	variableName	AvailabilityState	
	variableAttributes	mutability	ReadOnly
	variableCharacteristics	dataType	optionList
		valuesList	Available, Occupied, Reserved, Unavailable, Faulted
Description	<p>This variable reports current availability state for the ChargingStation and EVSE. If a Connector has its own availability state independent of the EVSE, then this variable may be used to report the Connector's availability state. As such it replicates ConnectorStatus values reported in StatusNotification messages.</p> <p>An EVSE component is addressed on its own tier. So, EVSE #1 is addressed as component EVSE on tier "evse = 1.</p>		

2.13.3. AllowReset

Required	no		
Component	componentName	EVSE	
		*	
Variable	variableName	AllowReset	
	variableAttributes	mutability	ReadOnly
	variableCharacteristics	dataType	boolean
Description	Component can be reset. Can be used to announce that an EVSE can be reset individually.		

2.13.4. ConnectorType

Required	yes		
Component	componentName	Connector	
		*	
Variable	variableName	ConnectorType	
	variableAttributes	mutability	ReadOnly
	variableCharacteristics	dataType	string
Description	Value of the type of connector as defined by ConnectorEnumType in "Part 2 - Specification" plus additionally: cGBT, cChaoJi, OppCharge.		

2.13.5. PhaseRotation

Required	no		
Component	componentName	*	
		*	
Variable	variableName	PhaseRotation	
	variableAttributes	mutability	ReadOnly or ReadWrite.
	variableCharacteristics	dataType	String

Description	This variable describes the phase rotation of a Component relative to its parent Component, using a three letter string consisting of the letters: R, S, T and x. The letter 'R' can be identified as phase 1 (L1), 'S' as phase 2 (L2), 'T' as phase 3 (L3). The lower case 'x' is used to designate a phase that is not connected. An empty string means that phase rotation is not applicable or not known. Certain measurands, like voltage and current, are reported with a phase relative to the grid connection. In order to support this, all components in the chain from Connector to ElectricalFeed need to have a value for PhaseRotation. Some examples: "" (unknown) "RST" (Standard Reference Phasing) "RTS" (Reversed Reference Phasing) "SRT" (Reversed 240 degree rotation) "STR" (Standard 120 degree rotation) "TRS" (Standard 240 degree rotation) "TSR" (Reversed 120 degree rotation) "RSx" (Two phases connected) "Rxx" (One phase connected)
--------------------	---

2.13.6. SupplyPhases

Required	yes		
Components	componentName	ChargingStation	
		EVSE	
		Connector	
	evse	* (for EVSE and Connector)	
Variable	variableName	SupplyPhases	
	variableAttributes	mutability	ReadOnly
	variableCharacteristics	dataType	integer
Description	Number of alternating current phases connected/available. 1 or 3 for AC, 0 means DC (no alternating phases). Null value indicates that the number of phases (e.g. in use) is unknown.		

2.13.7. Power

Required	yes (maxLimit only)		
Component	componentName	EVSE	
		*	
Variable	variableName	Power	
	variableAttributes	mutability	ReadOnly
	variableCharacteristics	dataType	decimal
		maxLimit	decimal
Description	The variableCharacteristic <i>maxLimit</i> , that holds the maximum power that this EVSE can provide, is required. The <i>Actual</i> value of the instantaneous (real) power is desired, but not required.		

2.13.8. Example Reporting of EVSEs and Connectors via device model

The following example illustrates how the device model reports EVSEs and Connectors for an example charging station that has two EVSEs, of which EVSE #1 has one Type2 connector and EVSE #2 has two connectors: CCS and CHAdeMO.

Component				Variable		VariableAttribute		VariableCharacteristics		
name	evse id	evse connectionId	instance	name	instance	type	value	dataType	maxLimit	supports Monitoring
ChargingStation				Available		Actual	true	boolean		false
ChargingStation				AvailabilityState		Actual	Available	boolean		false
ChargingStation				SupplyPhases		Actual	integer	3		false
ChargingStation				ACCurrent	"L1"	Actual	decimal	45.0		true
ChargingStation				ACCurrent	"L2"	Actual	decimal	44.9		true
ChargingStation				ACCurrent	"L3"	Actual	decimal	44.9		true
EVSE	1		"left"	Available		Actual	true	boolean		false
EVSE	1		"left"	AvailabilityState		Actual	Available	optionList		false
EVSE	1		"left"	SupplyPhases		Actual	3	integer		false
EVSE	1		"left"	Power		Actual	0.0	decimal	22000.0	true
Connector	1	1		Available		Actual	true	boolean		false
Connector	1	1		ConnectorType		Actual	sType2	string		false
Connector	1	1		SupplyPhases		Actual	3	integer		false
EVSE	2		"right"	Available		Actual	true	boolean		false
EVSE	2		"right"	AvailabilityState		Actual	Occupied	optionList		false
EVSE	2		"right"	SupplyPhases		Actual	0	integer		false
EVSE	2		"right"	Power		Actual	41000.0	decimal	50000.0	true
Connector	2	1		Available		Actual	true	boolean		false
Connector	2	1		AvailabilityState		Actual	Occupied	optionList		false
Connector	2	1		ConnectorType		Actual	cCCS2	string		false
Connector	2	1		SupplyPhases		Actual	0	integer		false
Connector	2	2		Available		Actual	true	boolean		false
Connector	2	2		AvailabilityState		Actual	Unavailable	optionList		false
Connector	2	2		ConnectorType		Actual	cG105	string		false
Connector	2	2		SupplyPhases		Actual	0	integer		false

NOTE

An instance name has been given to the EVSEs in this example. This is to illustrate that it is allowed to provide an instance name even if only one instance of the component exists. It is not required to do so.

The variable Voltage of ChargingStation has been added to show an example of a multi-instance variable. Not all VariableAttributes and VariableCharacteristics are shown in the table.

2.14. ISO 15118 Related

2.14.1. CentralContractValidationAllowed

Required	no			
Component	componentName	ISO15118Ctrlr		
Variable	variableName	CentralContractValidationAllowed		
	variableAttributes	mutability	ReadWrite	
	variableCharacteristics	dataType	boolean	
Description	If this variable exists and has the value <i>true</i> , then Charging Station can provide a contract certificate that it cannot validate, to the CSMS for validation as part of the AuthorizeRequest.			

2.14.2. ContractValidationOffline

Required	yes		
Component	componentName	ISO15118Ctrlr	

Variable	variableName	ContractValidationOffline	
	variableAttributes	mutability	ReadWrite
	variableCharacteristics	dataType	boolean
Description	If this variable is <i>true</i> , then Charging Station will try to validate a contract certificate when it is offline.		

2.14.3. ProtocolSupportedByEV

Required	no		
Component	componentName	ConnectedEV	
	evse	*	
Variable	variableName	ProtocolSupportedByEV	
	variableInstance	<Priority>	
	variableAttributes	mutability	ReadWrite
	variableCharacteristics	dataType	string
Description	<p>A string with the following comma-separated items: "<uri>,<major>,<minor>". This is information from the supportedAppProtocolReq message from ISO 15118. Each priority is given its own variable instance. Example: "urn:iso:15118:2:2013:MsgDef,2,0"</p>		

2.14.4. ProtocolAgreed

Required	no		
Component	componentName	ConnectedEV	
	evse	*	
Variable	variableName	ProtocolAgreed	
	variableAttributes	mutability	ReadWrite
	variableCharacteristics	dataType	string
	<p>A string with the following comma-separated items: "<uri>,<major>,<minor>". This is the protocol uri and version information that was agreed upon between EV and EVSE in the supportedAppProtocolReq handshake from ISO 15118. Example: "urn:iso:15118:2:2013:MsgDef,2,0"</p>		

2.14.5. ISO15118PnCEnabled

Required	no		
Component	componentName	ISO15118Ctrlr	
Variable	variableName	PnCEnabled	
	variableAttributes	mutability	ReadWrite
	variableCharacteristics	dataType	boolean
Description	<p>If this variable is <i>true</i>, then ISO 15118 plug and charge as described by use case C07 - Authorization using Contract Certificates is enabled. If this variable is <i>false</i>, then ISO 15118 plug and charge as described by use case C07 - Authorization using Contract Certificates is disabled.</p>		

2.14.6. ISO15118V2GCertificateInstallationEnabled

Required	no		
Component	componentName	ISO15118Ctrlr	

Variable	variableName	V2GCertificateInstallationEnabled	
	variableAttributes	mutability	ReadWrite
	variableCharacteristics	dataType	boolean
Description	If this variable is <i>true</i> , then ISO 15118 V2G Charging Station certificate installation as described by use case A02 - Update Charging Station Certificate by request of CSMS and A03 - Update Charging Station Certificate initiated by the Charging Station is enabled. If this variable is <i>false</i> , then ISO 15118 V2G Charging Station certificate installation as described by use case A02 - Update Charging Station Certificate by request of CSMS and A03 - Update Charging Station Certificate initiated by the Charging Station is disabled.		

2.14.7. ISO15118ContractCertificateInstallationEnabled

Required	no		
Component	componentName	ISO15118Ctrlr	
Variable	variableName	ContractCertificateInstallationEnabled	
	variableAttributes	mutability	ReadWrite
	variableCharacteristics	dataType	boolean
Description	If this variable is <i>true</i> , then ISO 15118 contract certificate installation/update as described by use case M01 - Certificate installation EV and M02 - Certificate Update EV is enabled. If this variable is <i>false</i> , then ISO 15118 contract certificate installation/update as described by use case M01 - Certificate installation EV and M02 - Certificate Update EV is disabled.		

2.14.8. ISO15118RequestMeteringReceipt

Required	no		
Component	componentName	ISO15118Ctrlr	
Variable	variableName	RequestMeteringReceipt	
	variableAttributes	mutability	ReadWrite
	variableCharacteristics	dataType	boolean
Description	If this variable is <i>true</i> , then Charging Station shall request a metering receipt from EV before sending a fiscal meter value to CSMS.		

2.14.9. ISO15118SeccId

Required	no		
Component	componentName	ISO15118Ctrlr	
	evse	* (optional)	
Variable	variableName	SeccId	
	variableAttributes	mutability	ReadWrite
	variableCharacteristics	dataType	string
Description	The name of the SECC in the string format as required by ISO 15118. It is used as the commonName (CN) of the SECC leaf certificate. Example: "DE-ICE-S-0003C4D5578786756453309675436-2"		

2.14.10. ISO15118CountryName

Required	no		
Component	componentName	ISO15118Ctrlr	
	evse	* (optional)	
Variable	variableName	CountryName	
	variableAttributes	mutability	ReadWrite
	variableCharacteristics	dataType	string

Description	The countryName of the SECC in the ISO 3166-1 format. It is used as the countryName (C) of the SECC leaf certificate. Example: "DE"	
--------------------	--	--

2.14.11. ISO15118OrganizationName

Required	no		
Component	componentName	ISO15118Ctrlr	
	evse	* (optional)	
Variable	variableName	OrganizationName	
	variableAttributes	mutability	ReadWrite
	variableCharacteristics	dataType	string
Description	<p>The organizationName of the CSO operating the charging station. It is used as the organizationName (O) of the SECC leaf certificate. Example: "John Doe Charging Services Ltd" Note: This value will usually be identical to SecurityCtrlr.OrganizationName, but it does not have to be.</p>		

2.14.12. ISO15118EvseId

Required	no		
Component	componentName	EVSE	
	evse	*	
Variable	variableName	ISO15118Evseld	
	variableAttributes	mutability	ReadWrite
	variableCharacteristics	dataType	string
Description	<p>The name of the EVSE in the string format as required by ISO 15118 and IEC 63119-2. Example: "DE*ICE*E*1234567890*1"</p>		



OCPP 2.0.1

Part 4 - JSON over WebSockets implementation guide

FINAL, 2020-03-31

Table of Contents

Disclaimer	1
Version History	2
1. Introduction	3
1.1. Purpose of this document	3
1.2. Intended audience	3
1.3. OCPP-S and OCPP-J	3
1.4. Conventions	3
1.5. Definitions & Abbreviations	3
1.6. References	4
2. Benefits & Issues	5
3. WebSockets	6
3.1. Client request	6
3.2. Server response	7
3.3. WebSocket Compression	8
4. RPC framework	9
4.1. Introduction	9
4.2. Message structures for different message types	11
4.3. RPC Framework Error Codes	14
4.4. Extension fallback mechanism	14
5. Connection	15
5.1. Data integrity	15
5.2. WebSocket Ping in relation to OCPP Heartbeat	15
5.3. Reconnecting	16
5.4. Network node hierarchy	16
6. OCPP Routing	17
6.1. Local Controller	17
6.2. Connections	17
6.3. Connection loss	18
6.4. Local Controller initiated messages	18
6.5. Local Controller Security	18
7. Signed Messages	20
7.1. Signed Message Format	20
7.2. Handling Signed Messages	20
7.3. Allowed Algorithms	20
7.4. Key Management	21
8. Configuration	22
8.1. RetryBackOffRepeatTimes	22
8.2. RetryBackOffRandomRange	22
8.3. RetryBackOffWaitMinimum	22
8.4. WebSocketPingInterval	22
9. CustomData Extension	24

Disclaimer

Copyright © 2010 – 2020 Open Charge Alliance. All rights reserved.

This document is made available under the **Creative Commons Attribution-NoDerivatives 4.0 International Public License** (<https://creativecommons.org/licenses/by-nd/4.0/legalcode>).

Version History

Version	Date	Author	Description
2.0.1	2020-03-31	Franc Buve (OCA) Milan Jansen (OCA) Robert de Leeuw (<i>iHomer</i>)	Final version of OCPP 2.0.1
2.0	2018-04-11	Milan Jansen (OCA) Robert de Leeuw (<i>iHomer</i>)	OCPP 2.0 April 2018 Update for 2.0 Section added for OCPP Routing and Signed Messages
1.6	2015-10-08	Patrick Rademakers (<i>iHomer</i>) Reinier Lamers (<i>The New Motion</i>) Robert de Leeuw (<i>iHomer</i>)	Updated to 1.6 Asciidoc formatting, remove JSON schema for 1.5 Some clarification Added 1.6 json schema

1. Introduction

1.1. Purpose of this document

The purpose of this document is to give the reader the information required to create a correct interoperable OCPP [JSON](#) implementation (OCPP-J). We will try to explain what is mandatory, what is considered good practice and what one should not do, based on our own experience. Undoubtedly misunderstandings or ambiguities will remain but by means of this document we aim to prevent them as much as possible.

1.2. Intended audience

This document is intended for developers looking to understand and/or implement OCPP JSON in a correct and interoperable way. Rudimentary knowledge of implementing web services on a server or embedded device is assumed.

1.3. OCPP-S and OCPP-J

With the introduction of OCPP 1.6, there were two different flavors of OCPP; SOAP and JSON. To avoid confusion in communication on the type of implementation we recommend using the distinct suffixes -J and -S to indicate JSON or SOAP. In generic terms this would be OCPP-J for JSON and OCPP-S for SOAP. Version specific terminology would be OCPP1.6J or OCPP1.2S. If no suffix is specified for OCPP 1.2 or 1.5 then a SOAP implementation must be assumed. As of release 1.6 this can no longer be implicit and should always be made clear. If a system supports both the JSON and SOAP variant it is considered good practice to label this OCPP1.6JS instead of just OCPP1.6.

OCPP 2.0.1 only supports JSON, but it is preferable to keep using the -J designation. As a new transport mechanism might be introduced in a future version of OCPP. So it will be OCPP2.0.1J.

1.4. Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [\[RFC2119\]](#).

1.5. Definitions & Abbreviations

Abbreviation	Description
IANA	Internet Assigned Numbers Authority (www.iana.org).
OCPP-J	OCPP communication over WebSocket using JSON. Specific OCPP versions should be indicated with the J extension. OCPP2.0.1J means we are talking about a JSON/WebSocket implementation of 2.0.1.
OCPP-S	OCPP communication over SOAP and HTTP(S). As of version 1.6 this should explicitly mentioned. Older versions are assumed to be S unless clearly specified otherwise, e.g. OCPP1.5 is the same as OCPP1.5S
RPC	Remote procedure call
WAMP	WAMP is an open WebSocket subprotocol that provides messaging patterns to handle asynchronous data.

1.6. References

Reference	Description
[EMI3-BO]	"eMI3 standard version V1.0" http://emi3group.com/documents-links/
[OCPP2.0-PART2]	"OCPP 2.0.1: Part 2 - Specification". http://www.openchargealliance.org/downloads/
[RFC1951]	"DEFLATE Compressed Data Format Specification version 1.3". https://www.ietf.org/rfc/rfc1951
[RFC2119]	"Key words for use in RFCs to Indicate Requirement Levels". S. Bradner. March 1997. http://www.ietf.org/rfc/rfc2119.txt
[RFC2616]	"Hypertext Transfer Protocol – HTTP/1.1". http://tools.ietf.org/html/rfc2616
[RFC2617]	"HTTP Authentication: Basic and Digest Access Authentication". http://tools.ietf.org/html/rfc2617
[RFC3629]	"UTF-8, a transformation format of ISO 10646". http://tools.ietf.org/html/rfc3629
[RFC3986]	"Uniform Resource Identifier (URI): Generic Syntax". http://tools.ietf.org/html/rfc3986
[RFC5246]	"The Transport Layer Security (TLS) Protocol; Version 1.2". http://tools.ietf.org/html/rfc5246
[RFC6455]	"The WebSocket Protocol". http://tools.ietf.org/html/rfc6455
[RFC7515]	"JSON Web Signatures (JWS)". https://tools.ietf.org/html/rfc7515
[RFC7518]	"JSON Web Algorithms (JWA)". https://tools.ietf.org/html/rfc7518
[RFC7692]	"Compression Extensions for WebSocket". https://tools.ietf.org/html/rfc7692
[RFC8259]	"The JavaScript Object Notation (JSON) Data Interchange Format". T. Bray. December 2017. https://tools.ietf.org/html/rfc8259
[WAMP]	http://wamp.ws/

2. Benefits & Issues

The WebSocket protocol is defined in [\[RFC6455\]](#). Working implementations of earlier draft WebSocket specifications exist, but OCPP-J implementations SHOULD use the protocol described in [\[RFC6455\]](#).

Be aware that WebSocket defines its own message structure on top of TCP. Data sent over a WebSocket, on a TCP level, is wrapped in a WebSocket frame with a header. When using a framework this is completely transparent. When working for an embedded system however, WebSocket libraries may not be available and then one has to frame messages correctly according to [\[RFC6455\]](#) him/herself.

3. WebSockets

For the connection between a Charging Station and a Charging Station Management System (CSMS) using OCPP-J, the CSMS acts as a WebSocket server and the Charging Station acts as a WebSocket client.

3.1. Client request

To set up a connection, the Charging Station initiates a WebSocket connection as described in [\[RFC6455\]](#) section 4, "Opening Handshake".

OCPP-J imposes extra constraints on the URL and the WebSocket subprotocol, detailed in the following two sections 4.1.1 and 4.1.2.

The Client (Charging Station) SHALL keep this WebSocket connection open all the time.

3.1.1. The connection URL

To initiate a WebSocket connection, the Charging Station needs a URL ([\[RFC3986\]](#)) to connect to. This URL is henceforth called the "connection URL". This connection URL is specific to a Charging Station. The Charging Station's connection URL contains the Charging Station identity so that the CSMS knows which Charging Station a WebSocket connection belongs to. However it is RECOMMENDED to let the CSMS NOT solely rely on the connection URL to identify a Charging Station, but to double-check the Charging Station's identity against their authentication credentials.

A CSMS supporting OCPP-J MUST provide at least one OCPP-J endpoint URL, from which the Charging Station SHOULD derive its connection URL. This OCPP-J endpoint URL can be any URL with a "ws" or "wss" scheme. How the Charging Station obtains an OCPP-J endpoint URL is outside of the scope of this document.

To derive its connection URL, the Charging Station modifies the OCPP-J endpoint URL by appending to the path first a '/' (U+002F SOLIDUS) and then a string uniquely identifying the Charging Station. This uniquely identifying string has to be percent-encoded as necessary as described in [\[RFC3986\]](#).

Example 1: for a Charging Station with identity "CS001" connecting to a CSMS with OCPP-J endpoint URL "ws://csms.example.com/ocpp" this would give the following connection URL:

`ws://csms.example.com/ocpp/CS001`

Example 2: for a Charging Station with identity "RDAM 123" connecting to a CSMS with OCPP-J endpoint URL "wss://csms.example.com/ocppj" this would give the following URL:

`wss://csms.example.com/ocppj/RDAM%20123`

The Charging Station identity datatype is *identifierString* (For definition see [\[OCPP2.0.1-PART2\]](#)) Additionally the colon ":" character might not be used, because the unique identifier is also used for the basic authentication username. The colon ":" character is used to separate the basic authentication username and the password. The maximum length of the Charging Station identity is: 48 (Note: Maximum length was chosen to ensure compatibility with EVSE ID from [\[EMI3-BO\]](#) "Part 2: business objects."

3.1.2. OCPP version

The OCPP version(s) MUST be specified in the Sec-Websocket-Protocol field. This SHOULD be one or more of the following values:

Table 1. OCPP Versions

OCPP version	WebSocket subprotocol name
1.2	ocpp1.2
1.5	ocpp1.5
1.6	ocpp1.6
2.0	ocpp2.0
2.0.1	ocpp2.0.1

The ones for OCPP 1.2, 1.5, 1.6, 2.0 and 2.0.1 are official WebSocket subprotocol name values. They are registered as such with IANA.

Note that OCPP 1.2 and 1.5 are in the list. Since the JSON over WebSocket solution is independent of the actual message content

the solution can be used for older OCPP versions as well. Please keep in mind that in these cases the implementation should preferably also maintain support for the SOAP based solution to be interoperable.

It is considered good practice to include the OCPP version as part of the OCPP-J endpoint URL string. If you run a web service that can handle multiple protocol versions on the same OCPP-J endpoint URL this is not necessary of course.

3.1.3. Example of an opening HTTP request

The following is an example of an opening HTTP request of an OCPP-J connection handshake:

```
GET /webServices/ocpp/CS3211 HTTP/1.1
Host: some.server.com:33033
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: x3JJHMBDL1EzLkh9GBhXDw==
Sec-WebSocket-Protocol: ocpp2.0.1, ocpp1.6
Sec-WebSocket-Version: 13
```

The bold parts are found as such in every WebSocket handshake request, the other parts are specific to this example.

In this example, the CSMS's OCPP-J endpoint URL is "ws://some.server.com:33033/webServices/ocpp". The Charging Station's unique identifier is "CS3211", so the path to request becomes "webServices/ocpp/CS3211".

With the Sec-WebSocket-Protocol header, the Charging Station indicates here that it can use OCPP2.0.1J and OCPP1.6J, with a preference for the former.

The other headers in this example are part of the HTTP and WebSocket protocols and are not relevant to those implementing OCPP-J on top of third-party WebSocket libraries. The roles of these headers are explained in [\[RFC2616\]](#) and [\[RFC6455\]](#).

3.2. Server response

Upon receiving the Charging Station's request, the CSMS has to finish the handshake with a response as described in [\[RFC6455\]](#).

The following OCPP-J-specific conditions apply:

- If the CSMS does not recognize the Charging Station identifier in the URL path, it SHOULD send an HTTP response with status 404 and abort the WebSocket connection as described in [\[RFC6455\]](#).
- If the CSMS does not agree to using one of the subprotocols offered by the client, it MUST complete the WebSocket handshake with a response without a Sec-WebSocket-Protocol header and then immediately close the WebSocket connection.

So if the CSMS accepts the above example request and agrees to using OCPP 2.0.1J with the Charging Station, the CSMS's response will look as follows:

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo=
Sec-WebSocket-Protocol: ocpp2.0.1
```

The bold parts are found as such in every WebSocket handshake response, the other parts are specific to this example.

The role of the Sec-WebSocket-Accept header is explained in [\[RFC6455\]](#).

The Sec-WebSocket-Protocol header indicates that the server will be using OCPP2.0.1J on this connection.

For a definition on how a server SHALL report a 'failure to process the message', see: [CALLERROR](#)

3.3. WebSocket Compression

OCPP 2.0.1 supports RFC 7692: Compression Extensions for WebSocket see: [\[RFC6455\]](#)

Table 2. WebSocket compression support requirement for devices

Device	WebSocket Compression Support
Charging Station	Optional
CSMS	Required
Local Controller	Required

OCPP Requires the CSMS (and Local Controller) to support [RFC 7692](#), WebSocket compression is seen as a relative simple way to reduce mobile data usage. For a Charging Station this is not a hard requirement, as this might be more complex to implement on an embedded platform, but as this is seen as efficient solution to reduce mobile data usage, it is RECOMMENDED to be implemented on a Charging Station that uses a mobile data connection.

OCPP Requires the CSMS (and Local Controller) to support [RFC 7692](#), WebSocket compression is seen as a relative simple way to reduce mobile data usage. For a Charging Station this is not a hard requirement, as this might be more complex to implement on an embedded platform, but as this is seen as efficient solution to reduce mobile data usage, it is RECOMMENDED to be implemented on a Charging Station that uses a mobile data connection.

[RFC 7692](#) allows the Charging Station and the CSMS to do a negotiation during the connection setup. When both parties support the Compression Extension they will then use DEFLATE compression ([\[RFC1951\]](#)) when sending data over the line. When one of the parties doesn't support it, the JSON will be sent uncompressed (like in OCPP 1.6J).

When the Charging Station detects that compression is not used, it is RECOMMENDED not to close the connection, as turning off compression can be very useful during development, testing and debugging.

For more detailed information read the [RFC 7692](#).

4. RPC framework

4.1. Introduction

A websocket is a full-duplex connection, simply put a pipe where data goes in and data can come out and without a clear relation between in and out. The WebSocket protocol by itself provides no way to relate messages as requests and responses. To encode these request/response relations we need a small protocol on top of WebSocket. This problem occurs in more use cases of WebSocket so there are existing schemes to solve it. The most widely-used is WAMP (see [WAMP]) but with the current version of that framework handling RPCs symmetrically is not WAMP compliant. Since the required framework is very simple we decided to define our own framework, inspired by WAMP, leaving out what we do not need and adding what we find missing.

Basically what we need is very simple: we need to send a message (CALL) and receive a reply (CALLRESULT) or an explanation why the message could not be handled properly (CALLERROR). For possible future compatibility we will keep the numbering of these message in sync with WAMP. Our actual OCPP message will be put into a wrapper that at least contains the type of message, a unique message ID and the payload, the OCPP message itself.

4.1.1. Synchronicity

A Charging Station or CSMS SHALL NOT send a CALL message to the other party unless all the CALL messages it sent before have been responded to or have timed out. This does not mean that the CSMS cannot send a message to another Charging Station, while waiting for a response of a first Charging Station, this rule is per OCPP-J connection. A CALL message has been responded to when a CALLERROR or CALLRESULT message has been received with the message ID of the CALL message.

A CALL message has timed out when:

- it has not been responded to, and
- an implementation-dependent timeout interval has elapsed since the message was sent.

Implementations are free to choose this timeout interval. It is RECOMMENDED that they take into account the kind of network used to communicate with the other party. Mobile networks typically have much longer worst-case round-trip times than fixed lines.

NOTE

The above requirements do not rule out that a Charging Station or CSMS will receive a CALL message from the other party while it is waiting for a CALLERROR or CALLRESULT. Such a situation is difficult to prevent because CALL messages from both sides can always cross each other.

4.1.2. Message validity and Character encoding

The whole message consisting of wrapper and payload MUST be valid JSON encoded with the UTF-8 (see [RFC3629]) character encoding. Furthermore, the Charging Station and CSMS have the right to reject messages which are not conform the JSON schemas.

Note that all valid US-ASCII text is also valid UTF-8, so if a system sends only US-ASCII text, all messages it sends comply with the UTF-8 requirement. Non US-ASCII characters SHOULD only be used for sending natural-language text. An example of such natural-language text is the MessageType which contains the text of a DisplayMessage in OCPP 2.0.1

4.1.3. The message type

To identify the type of message one of the following Message Type Numbers MUST be used.

Table 3. Message types

MessageType	Message Type Number	Description
CALL	2	Request message
CALLRESULT	3	Response message
CALLERROR	4	Error response to a request message

When a server receives a message with a Message Type Number not in this list, it SHALL ignore the message payload. Each message type may have additional required fields.

4.1.4. The message ID

The message ID serves to identify a request. A message ID for any CALL message MUST be different from all message IDs previously used by the same sender for any other CALL messages on the same WebSocket connection. A message ID for a CALLRESULT or CALLERROR message MUST be equal to that of the CALL message that the CALLRESULT or CALLERROR message is a response to.

Table 4. Unique Message ID

Name	Datatype	Restrictions
messageld	string[36]	Unique message ID, maximum length of 36 characters, to allow for UUIDs/GUIDs

4.1.5. JSON Payload

The Payload of a message is a JSON object containing the arguments relevant to the Action.

If there is no payload JSON allows for two different notations: `null` or and empty object `{}`. Although it seems trivial, we consider it good practice to only use the empty object statement. Null usually represents something undefined, which is not the same as empty, and also `{}` is shorter.

When a field is optional in the OCPP action (0..1 or 0..*) and is left empty for a specific request/response, JSON allows for a couple of different ways to put this in a JSON string. But because OCPP is designed for wireless links, OCPP only allows 1 option: Do not put the field in the payload (so `null`, `{}` or `[]` are not allowed for an empty field).

When a field has a cardinality of zero/one to many (0..* or 1..*) and it has been given one entity, than it will still remain a list, but of size 1.

4.1.6. Action

The Action field in the `CALL` message MUST be the OCPP message name without the "Request" suffix.

For example: For a "BootNotificationRequest" the action field will be set to "BootNotification".

BTW: The `CALLRESULT` does not contain the action field. A client can match the Response (`CALLRESULT`) with the Request (`CALL`) via the Messageld field.

4.1.7. Message Validity

An message is only valid when:

- Action is a known Action.
- The JSON payload is valid JSON
- All the required field for the Action are present
- All data is of the correct data type.

When a message is not valid, the server SHALL respond with a CALLERROR

4.2. Message structures for different message types

NOTE

You may find the Charging Station identity missing in the following paragraphs. The identity is exchanged during the WebSocket connection handshake and is a property of the connection. Every message is sent by or directed at this identity. There is therefore no need to repeat it in each message.

4.2.1. CALL

A CALL always consists of 4 elements: The standard elements MessageTypeld and Messageld, a specific Action that is required on the other side and a payload, the arguments to the Action. The syntax of a CALL looks like this:

[<MessageTypeld>, "<Messageld>", "<Action>", {<Payload>}]

Table 5. CALL Fields

Field	Datatype	Meaning
MessageTypeld	integer	This is a Message Type Number which is used to identify the type of the message.
Messageld	string[36]	This is a unique identifier that will be used to match request and result.
Action	string	The name of the remote procedure or action. This field SHALL contain a case-sensitive string. The field SHALL contain the OCPP Message name without the "Request" suffix. For example: For a "BootNotificationRequest", this field shall be set to "BootNotification".
Payload	JSON	JSON Payload of the action, see: JSON Payload for more information.

For example, a BootNotificationRequest could look like this:

```
[ 2,
  "19223201",
  "BootNotification",
  {
    "reason": "PowerUp",
    "chargingStation": {
      "model": "SingleSocketCharger",
      "vendorName": "VendorX"
    }
  }
]
```

4.2.2. CALLRESULT

If the call can be handled correctly the result will be a regular CALLRESULT. Error situations that are covered by the definition of the OCPP response definition are not considered errors in this context. They are regular results and as such will be treated as a normal CALLRESULT, even if the result is undesirable for the recipient.

A CALLRESULT always consists of 3 elements: The standard elements MessageTypeld and Messageld and a payload, containing the response to the Action in the original Call.

The syntax of a CALLRESULT looks like this:

`[<MessageTypeId>, "<MessageId>", {<Payload>}]`

Table 6. CALLRESULT Fields

Field	Datatype	Meaning
MessageTypeId	integer	This is a Message Type Number which is used to identify the type of the message.
MessageId	string[36]	This must be the exact same ID that is in the call request so that the recipient can match request and result.
Payload	JSON	JSON Payload of the action, see: JSON Payload for more information.

For example, a BootNotification response could look like this:

```
[ 3,
  "19223201",
  {
    "currentTime": "2013-02-01T20:53:32.486Z",
    "interval": 300,
    "status": "Accepted"
  }
]
```

4.2.3. CALLERROR

We only use CALLERROR in two situations:

1. An error occurred during the transport of the message. This can be a network issue, an availability of service issue, etc.
2. The call is received but the content of the call does not meet the requirements for a proper message. This could be missing mandatory fields, an existing call with the same unique identifier is being handled already, unique identifier too long, invalid JSON or OCPP syntax etc.

When a server needs to report a 'failure to process the message', the server SHALL use a Message Type: CallError (MessageTypeNumber = 4).

When a server receives a corrupt message, the CALLERROR SHALL NOT directly include syntactically invalid JSON (For example, without encoding it first). When also the Messageld cannot be read, the CALLERROR SHALL contain "-1" as Messageld.

When a message contains any invalid OCPP and/or it is not conform the JSON schema, the system is allowed to drop the message.

A CALLERROR always consists of 5 elements: The standard elements MessageTypeId and Messageld, an errorCode string, an errorDescription string and an errorDetails object.

The syntax of a CALLERROR looks like this:

```
[<MessageTypeId>, "<Messageld>", "<errorCode>", "<errorDescription>", {<errorDetails>}]
```

Table 7. CALLERROR Fields

Field	Datatype	Meaning
MessageTypeId	integer	This is a Message Type Number which is used to identify the type of the message.
Messageld	string[36]	This must be the exact same id that is in the call request so that the recipient can match request and result.
ErrorCode	string	This field must contain a string from the RPC Framework Error Codes table .
ErrorDescription	string[255]	Should be filled in if possible, otherwise a clear empty string "".
ErrorDetails	JSON	This JSON object describes error details in an undefined way. If there are no error details you MUST fill in an empty object {}.

For example, a CALLERROR could look like this:

```
[ 4,
  "162376037",
  "NotSupported",
  "SetDisplayMessageRequest not implemented",
  {}
]
```

4.3. RPC Framework Error Codes

The following table contains all the allowed error codes for the OCPP RPC Framework.

Table 8. Valid Error Codes

Error Code	Description
FormatViolation	Payload for Action is syntactically incorrect
GenericError	Any other error not covered by the more specific error codes in this table
InternalError	An internal error occurred and the receiver was not able to process the requested Action successfully
MessageTypeNotSupported	A message with an Message Type Number received that is not supported by this implementation.
NotImplemented	Requested Action is not known by receiver
NotSupported	Requested Action is recognized but not supported by the receiver
OccurrenceConstraintViolation	Payload for Action is syntactically correct but at least one of the fields violates occurrence constraints
PropertyConstraintViolation	Payload is syntactically correct but at least one field contains an invalid value
ProtocolError	Payload for Action is not conform the PDU structure
RpcFrameworkError	Content of the call is not a valid RPC Request, for example: MessageId could not be read.
SecurityError	During the processing of Action a security issue occurred preventing receiver from completing the Action successfully
TypeConstraintViolation	Payload for Action is syntactically correct but at least one of the fields violates data type constraints (e.g. "somestring": 12)

4.4. Extension fallback mechanism

Future versions of OCPP might add extra Message Types (other then [CALL](#), [CALLRESULT](#) and [CALLERROR](#))

When an OCPP 2.0.1 implementation receives a message with an unknown message type, it SHALL respond with an [CALLERROR](#) with errorCode: [MessageTypeNotSupported](#). This SHOULD notify the sending party about the not supported Message Type. The Sending Party SHALL then either terminate the connection, or fallback to the known: [CALL](#), [CALLRESULT](#) and [CALLERROR](#).

5. Connection

5.1. Data integrity

For data integrity we rely on the underlying TCP/IP transport layer mechanisms.

5.2. WebSocket Ping in relation to OCPP Heartbeat

The WebSocket specification defines Ping and Pong frames that are used to check if the remote endpoint is still responsive. In practice this mechanism is also used to prevent the network operator from quietly closing the underlying network connection after a certain period of inactivity. This websocket feature can be used as a substitute for most of the OCPP Heartbeat messages, but cannot replace all of its functionality.

An important aspect of the Heartbeat response is time synchronisation. The Ping and Pong frames cannot be used for this so at least one original Heartbeat message a day is recommended to ensure a correct clock setting on the Charging Station.

5.3. Reconnecting

When the connection is lost, the Charging Station SHALL try to reconnect. When reconnecting, the Charging Station SHALL use an increasing back-off time with some randomization until it has successfully reconnected. This prevents an overload of the CSMS when all Charging Stations reconnect after a restart of the CSMS.

The first reconnection attempts SHALL be after a back-off time of: `RetryBackOffWaitMinimum` seconds, plus a random value with a maximum of `RetryBackOffRandomRange` seconds. After every failed reconnection attempt the Charging Station SHALL double the previous back-off time, with a maximum of `RetryBackOffRepeatTimes`, adding a new random value with a maximum of `RetryBackOffRandomRange` seconds to every reconnection attempt. After `RetryBackOffRepeatTimes` reconnection attempts, the Charging Station SHALL keep reconnecting with the last back-off time, not increasing it any further.

When reconnecting, a Charging Station should not send a BootNotification unless one or more of the elements in the BootNotification have changed since the last connection. For the previous SOAP based solutions this was considered good practice but when using WebSocket the server can already make the match between the identity and a communication channel at the moment the connection is established. There is no need for an additional message.

5.4. Network node hierarchy

The physical network topology is not influenced by a choice for JSON or SOAP. In case of JSON however the issues with Network Address Translation (NAT) have been resolved by letting the Charging Station open a TCP connection to the CSMS and keeping this connection open for communication initiated by the CSMS. It is therefore no longer necessary to have a smart device capable of interpreting and redirecting SOAP calls in between the CSMS and the Charging Station.

6. OCPP Routing

For some topologies it is required to route OCPP-J messages. For example when implementing a Local Controller.

This section contains a solution for OCPP message routing that will work with any Charging Station and CSMS.

6.1. Local Controller

A Local controller is a device that sits between the CSMS and any number of Charging Stations, creating a local group. It is located near to the Charging Station (maybe even connected wired to the Charging Stations), so it does not have problem of losing the connection to the Charging Stations. This is practically useful for doing Local Smart Charging: load balancing between the Charging Stations on the same location. The Local Controller can see all the messages, ongoing transactions etc. It can send charging profiles to the Charging Station to influence the energy used by the Charging Stations, this way preventing the group to use more energy than available at the location at that time.

The Local Controller SHALL work so the Charging Station doesn't have to behave different when connected to the Local Controller, compared to a direct connection to a CSMS. A Local Controller SHALL work so that a Charging Station can work out of the box with the Local Controller, requiring only the parameters that are needed to connect to the Local Controller to be set. The Local Controller SHALL work so that the CSMS can not notice if the Charging Station is connecting to it directly, or via the Local Controller.

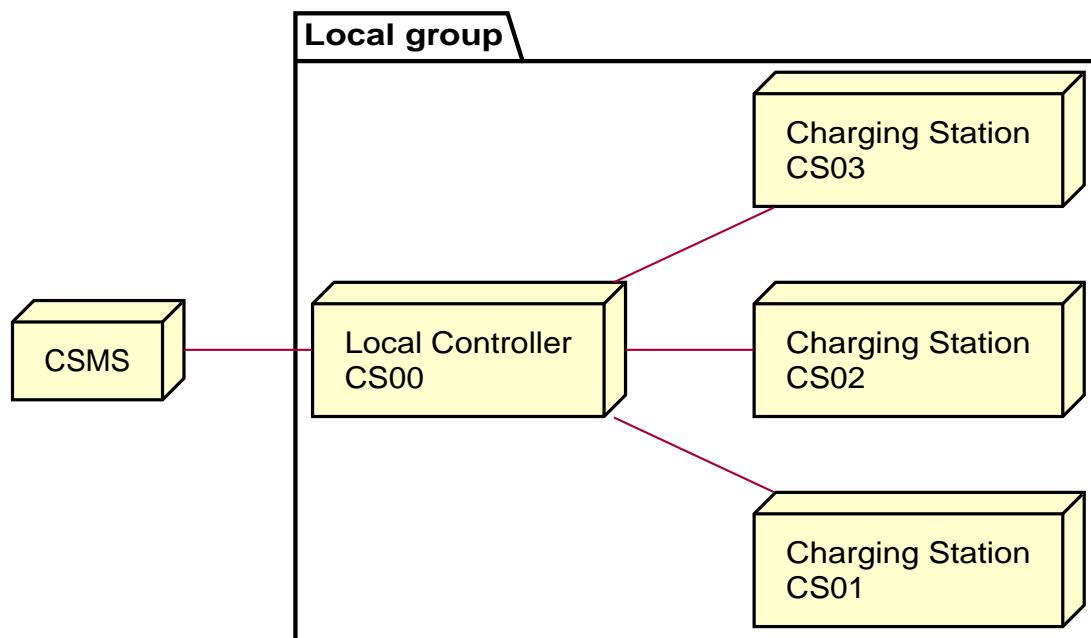


Figure 1. Local Controller Topology

6.2. Connections

For each Charging Station that connects to the Local Controller, two connections will be established:

1. A WebSocket connection from the Charging Station to the Local Controller (configured in the Charging Station)
2. A WebSocket connection from the Local Controller to the CSMS (configured in the Local Controller)

Both connections should use a similar connection URI with the same Charging Station identifier. To the CSMS, the connection from the Local Controller appears to be a regular Charging Station connection.

The Local Controller may open a separate WebSocket connection to the CSMS that allows the CSMS to address the Local Controller directly, which may be useful for changing settings or setting overall Charging Profiles.

When a Charging Station connects to the Local Controller, it SHALL connect to it like it would to a CSMS, using the same URI Path in the [connection URL](#) as it would use to connect to the CSMS. When the connection between Charging Station and the CSMS is successfully set up, the Local Controller SHALL set up a WebSocket connection to the CSMS with the same URI Path in the [connection URL](#) that was used by the Charging Station to setup the connection. The Local Controller SHALL open a WebSocket connection for every Charging Station that connects to it.

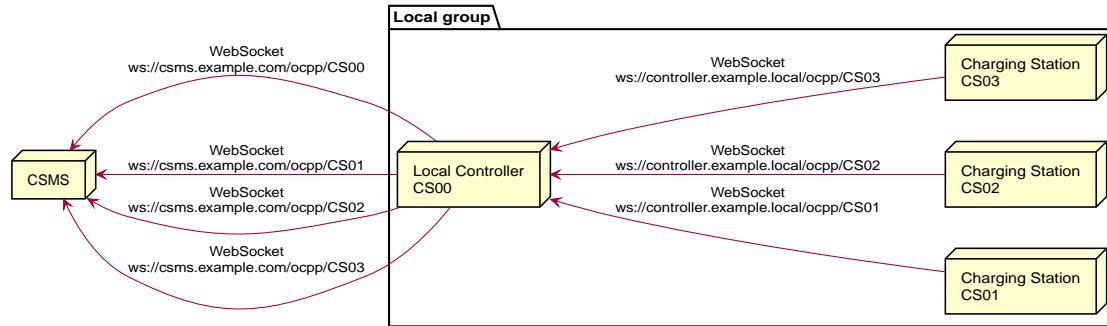


Figure 2. Local Controller WebSocket Connections

6.3. Connection loss

Whenever one or more WebSocket connections between CSMS and the Local Controller are lost, the Local Controller SHALL close all corresponding WebSockets to the Charging Stations that are connected to it. This is needed to force the Charging Station to queue messages as it would have done if it would have been connected to the CSMS directly and would have lost the connection to that CSMS.

Whenever the connection between a Charging Station and the Local Controller is lost, the Local Controller SHALL close the WebSocket connection it has for the Charging Station to the CSMS. This is needed to let the CSMS know the Charging Station is offline and no CSMS initiated messages can be sent to it.

6.4. Local Controller initiated messages

The Local Controller SHALL relay any Charging Station initiated message to the CSMS (and vice versa).

Since the Local Controller can also initiate its own messages to the Charging Station(s), a Local Controller SHALL take care of the following:

1. If a Local Controller sends its own messages to a Charging Station, it SHALL guarantee that its messages have IDs that do not collide with IDs used by the CSMS, now and in the future. This can be done by either assigning a range of numbers to the Local Controller to use (and the CSMS to skip), or by using UUIDs/GUIDs.
2. Replies to messages from the Charging Station to messages initiated by the Local Controller SHALL NOT be sent to the CSMS.

6.5. Local Controller Security

For the local controller, the normal OCPP security mechanisms will be used, as described in [OCPP2.0.1-PART2], part A. Security. All security profiles described there MAY be used when a Local Controller is deployed. The security section (part A) only describes the roles of the CSMS, and Charging Station. When a local controller is used, the security specification SHALL be interpreted as follows:

- In the connection from the Charging Station to the Local Controller, the Charging Station SHALL act as the Charging Station, and the Local Controller SHALL act as the CSMS. When TLS is used, the Local Controller SHALL be the TLS server, and the Charging Station SHALL be the TLS client.
- In the connection from the Local Controller to the CSMS, the Local Controller SHALL act as the Charging Station, and the CSMS SHALL act as the CSMS. When TLS is used, the CSMS SHALL be the TLS server, and the Local Controller SHALL be the TLS client.

When TLS with Client Side Certificates is used, the Local Controller SHALL have both a CSMS Certificate, and a Charging Station certificate (see [OCPP2.0.1-PART2] Part A - Keys used in OCPP), as it can function in both roles. These certificates SHALL be unique to the Local Controller. The Local Controller SHALL NOT store the Charging Station certificates of the attached Charging Stations. It SHALL also NOT store the CSMS Certificate of the CSMS. These certificates SHALL be kept private on their respective owners. The Local Controller SHALL only use its own certificates for setting up the TLS connections.

It SHALL be possible to distinguish the Local Controller from the CSMS based on the URL in the CSMS Certificate. Because the Local Controller is placed in the field, there is a risk that its certificates get stolen from it, e.g. by an attack on the hardware. In that case, it SHALL only be possible to use the CSMS Certificate on the Local Controller to communicate with the attached Charging Stations, not with any other Charging Stations in the infrastructure.

The TLS connections terminate on the Local Controller. So, the Local Controller can both read and manipulate data sent between the CSMSs and Charging Stations. If the security of the Local Controller is compromised, it will affect all attached Charging Stations. It is therefore RECOMMENDED to take sufficient security measures to protect the Local Controller. It is also

RECOMMENDED to sign critical commands or replies with the mechanism described in [Signed Messages](#). In this way, it can be detected if the Local Controller tries to manipulate data.

7. Signed Messages

For certain architectures it can be useful to use signed OCPP messages. This gives the Charging Station and the CSMS the ability to guarantee that messages are sent by the other party. For example when a Local Controller is involved, the Charging Station can know that a message received from the Local Controller is created and signed by the CSMS.

Message signing can also be used when forwarding data from the Charging Station or the CSMS to 3rd parties such as a DSO (Distribution System Operator).

Because message signing is not needed in all architectures and scenarios, it is not required for all OCPP implementations. It will depend on the security requirements if this is required.

This section defines a method to digitally sign any OCPP-J message. For each normal OCPP message an equivalent signed message is defined that encapsulates the normal message, and adds a digital signature.

7.1. Signed Message Format

For Signed OCPP Messages, JWS is used. For more information see: [\[RFC7515\]](#).

Suppose we have an OPC calls encoded in the OCPP-J message:

```
[<MessageTypeId>, "<MessageId>", {<Extension>}, "<Action>", {<Payload>}]
```

Then we define the equivalent signed message as follows.

```
[<MessageTypeId>, "<MessageId>", {<Extension>}, "<SignedAction>", {<SignedPayload>}]
```

The `MessageTypeId` and `MessageId` SHALL stay the same. The `<SignedAction>` field SHALL be the action name (`<Action>`) with the string "-Signed" appended. For instance, if "`<Action>`" is "BootNotification", then "`<SignedAction>`" is "BootNotification-Signed". The `<SignedPayload>` SHALL be the JWS encoding of the payload, computed according to the following settings:

- The JWS Payload SHALL be the `<Payload>` from the original message.
- The JWS Protected Header SHALL contain a field called "OCPPAction" containing the name (`<Action>`) of the OCPP action, and a field called "OCPPMessageTypedId" containing the message ID (`<MessageTypeId>`).
- The JWS Protected Header SHOULD contain the `x5t#S256` field to identify the key used for signing, as specified in Section 7.4.
- The `<SignedPayload>` SHALL be encoded using the Flattened JWS JSON Serialization syntax.

7.2. Handling Signed Messages

When a Charging Station or CSMS receives a signed message, it SHALL extract the encapsulated normal message. It SHALL process it normally, following the OCPP 2.0.1 standard. It MAY perform additional actions that use the digital signature. This is optional, because a secure connection between the CSMS and the Charging Station is expected, hence a second process to validate a signature on message level is redundant. When a Charging Station receives a signed request, and it supports digital signing, it SHALL send a signed reply.

7.3. Allowed Algorithms

The algorithms allowed for use with JSON Web Signatures are defined in the JSON Web Algorithms standard [\[RFC7518\]](#). To limit the cryptographic algorithms that a Charging Station has to implement, for OCPP the same algorithms SHALL be used as for the TLS connection used to secure communications. This means that for generating the digital signatures, the Charging Station and CSMS SHALL use the following algorithms from the JSON Web Algorithms standard [\[RFC7518\]](#), section 3.1:

- ES256: ECDSA using P-256 and SHA-256
- RS256: RSASSA-PKCS1-v1_5 using SHA-256
- RS384: RSASSA-PKCS1-v1_5 using SHA-384

Note that RS256 and ES256 are the algorithms recommended by [\[RFC7518\]](#).

7.4. Key Management

This section does not prescribe specific keys to be used for digital signatures. The CSMS Certificate and Charging Station Certificate, used for setting up a secure TLS connection, MAY be used for signing. For many use cases, these will however not be the correct keys. For instance, if the use case is to provide non-repudiation of meter readings, the messages should be signed with a certificate stored in the calibrated measuring chip.

To be able to verify the digital signature, one needs to know which key was used to sign it. JSON Web Signatures supports several ways to store a key identifier with the signed message. As the certificates that can be used for signing are not specified, hash values will be used to identify them. Within the OCPP, the Charging Station and CSMS SHOULD include the field `x5t#S256` in the JWS Protected Header to identify the certificate. Following the JSON Web Signatures standard [RFC7515] the value of this field SHOULD be set to the SHA-256 hash of the DER encoding of the signing certificate. How stakeholders can look up the certificate based on the hash value is out of scope for this document.

NOTE

In the set up with a Local Controller, described in [Local Controller](#), the TLS client key that would be signing messages to the CSMS will, in fact, not be the TLS client key that the TLS connection is using, since the key in the Local Controller is different from that in the Charging Station. Similarly, the TLS server key signing messages will be that of the CSMS, not that of the local controller. Therefore, implementation of the protocol MUST NOT regard this mismatch as invalidating the signatures; in fact, it is an expected and desired property to provide end-to-end authenticity.

8. Configuration

The following Configuration Variables are added to control JSON/WebSockets behaviour:

8.1. RetryBackOffRepeatTimes

Required	yes
Component	componentName OCPPCommCtrlr
Variable	variableName RetryBackOffRepeatTimes
	variableAttributes mutability ReadWrite
	variableCharacteristics dataType integer
Description	When the Charging Station is reconnecting, after a connection loss, it will use this variable for the amount of times it will double the previous back-off time. When the maximum number of increments is reached, the Charging Station keeps connecting with the same back-off time.

8.2. RetryBackOffRandomRange

Required	yes
Component	componentName OCPPCommCtrlr
Variable	variableName RetryBackOffRandomRange
	variableAttributes mutability ReadWrite
	variableCharacteristics unit seconds dataType integer
Description	When the Charging Station is reconnecting, after a connection loss, it will use this variable as the maximum value for the random part of the back-off time. It will add a new random value to every increasing back-off time, including the first connection attempt (with this maximum), for the amount of times it will double the previous back-off time. When the maximum number of increments is reached, the Charging Station will keep connecting with the same back-off time.

8.3. RetryBackOffWaitMinimum

Required	yes
Component	componentName OCPPCommCtrlr
Variable	variableName RetryBackOffWaitMinimum
	variableAttributes mutability ReadWrite
	variableCharacteristics unit seconds dataType integer
Description	When the Charging Station is reconnecting, after a connection loss, it will use this variable as the minimum back-off time, the first time it tries to reconnect.

8.4. WebSocketPingInterval

Required	yes
Component	componentName OCPPCommCtrlr
Variable	variableName WebSocketPingInterval
	variableAttributes mutability ReadWrite
	variableCharacteristics unit seconds dataType integer

Description	A value of 0 disables client side websocket Ping / Pong. In this case there is either no ping / pong or the server initiates the ping and client responds with Pong. Positive values are interpreted as number of seconds between pings. Negative values are not allowed, SetConfiguration is then expected to return a <i>Rejected</i> result. It is recommended to configure WebSocketPingInterval smaller than: MessageAttemptsTransactionEvent * MessageAttemptIntervalTransactionEvent. This will limit the chance of the resend mechanism for transaction-related messages being triggered by connectivity issues.
--------------------	---

9. CustomData Extension

In the JSON schema files all classes have the attribute `additionalProperties` set to `false`, such that a JSON parser will not accept any other properties in the message. In order to allow for some flexibility to create non-standard extensions for experimentation purposes, every JSON class has been extended with a "customData" property. This property is of type "CustomDataType", which has only one required property: "vendorId", which is used to identify the kind of customization. However, since it does not have `additionalProperties` set to `false` it can be freely extended with new properties.

In the same way as is defined for the DataTransfer message, the "vendorId" should be a value from the reversed DNS namespace, where the top tiers of the name, when reversed, should correspond to the publicly registered primary DNS name of the Vendor organization.

The following example shows the "CustomDataType" definition and the (optional) "customData" property in the schema definition of HeartbeatRequest:

```
{
    "$schema": "http://json-schema.org/draft-06/schema#",
    "$id": "HeartbeatRequest",
    "definitions": {
        "CustomDataType": {
            "description": "This class does not get 'AdditionalProperties = false' in the schema generation, so it can be extended with arbitrary JSON properties to allow adding custom data.",
            "javaType": "CustomData",
            "type": "object",
            "properties": {
                "vendorId": {
                    "type": "string",
                    "maxLength": 255
                }
            },
            "required": [ "vendorId" ]
        }
    },
    "type": "object",
    "additionalProperties": false,
    "properties": {
        "customData": {
            "$ref": "#/definitions/CustomDataType"
        }
    }
}
```

Whereas the standard HeartbeatRequest has an empty body, a customized version, that provides the value of the main meter and a count of all sessions to date, could look like this:

```
{
    "customData": {
        "vendorId": "com.mycompany.customheartbeat",
        "mainMeterValue": 12345,
        "sessionsToDate": 342
    }
}
```

A CSMS that has implemented this extension, identified by its "vendorId", will be able to process the data. Other CSMS

implementations will simply ignore these custom properties.

A CSMS can request a report of the *CustomizationCtrlr* component to get a list of all customizations that are supported by the Charging Station.



OCPP 2.0.1

Part 5 - Certification Profiles

Core & Advanced Security, FINAL, 2023-06-30

Table of Contents

1. Introduction & Reading Guide	2
2. Certification profiles	3
3. Features	5
3.1. Optional feature list for charging station.....	6
3.2. Optional feature list for CSMS	8
4. List of test cases	10
4.1. Introduction	10
4.2. Test Cases Core	10
4.3. Test Cases Advanced Security	31
4.4. Test Cases Local Authorization List Management	32
4.5. Test Cases Smart Charging	32
4.6. Test Cases Advanced Device Management	32
4.7. Test Cases Reservation	32
4.8. Test Cases Advanced User Interface.....	32
4.9. Test Cases ISO 15118 Support	32
5. OCPP 2.0.1 Mandatory Controller components per profile.....	33
6. Appendix A: additional questions for the Protocol Implementation Conformance Statement	34
6.1. Questions for Charging Stations	34
6.2. Questions for CSMSs.....	34
7. Appendix B: Hardware feature set	35
8. Appendix C: Features vs. OCPP use cases	36

Copyright © 2010 – 2023 Open Charge Alliance. All rights reserved.

This document is made available under the **Creative Commons Attribution-NoDerivatives 4.0 International Public License** (<https://creativecommons.org/licenses/by-nd/4.0/legalcode>).

1. Introduction & Reading Guide

This document describes the certification profiles for OCPP 2.0.1. These profiles are sets of use cases that can be certified via the Open Charge Alliance. This document contains the details on what is part of the OCPP 2.0.1 Certification. This document contains:

- The certification profiles and an overview of the functionality per profile.
- The list of optional features. This list contains specific functionality that is not mandatory for certification, but which can optionally be certified.
- The list of test cases for each of the certification profiles.
- The overview of the controller components that must be implemented per profile for certification testing.

For clarity: in the context of the OCPP Certification Program, the term *test case* refers to a sequence of messages for testing a use case from OCPP. The term *feature* refers to a functionality, that can be tested with one or more test cases (see [Features](#) for a more detailed explanation). Instead of making specific test cases mandatory or optional, the certification program for OCPP 2.0.1 works with features that are optional. Depending on whether the System Under Test (SUT) has implemented a feature, the test case(s) that belong to this feature, must be successfully passed or not.

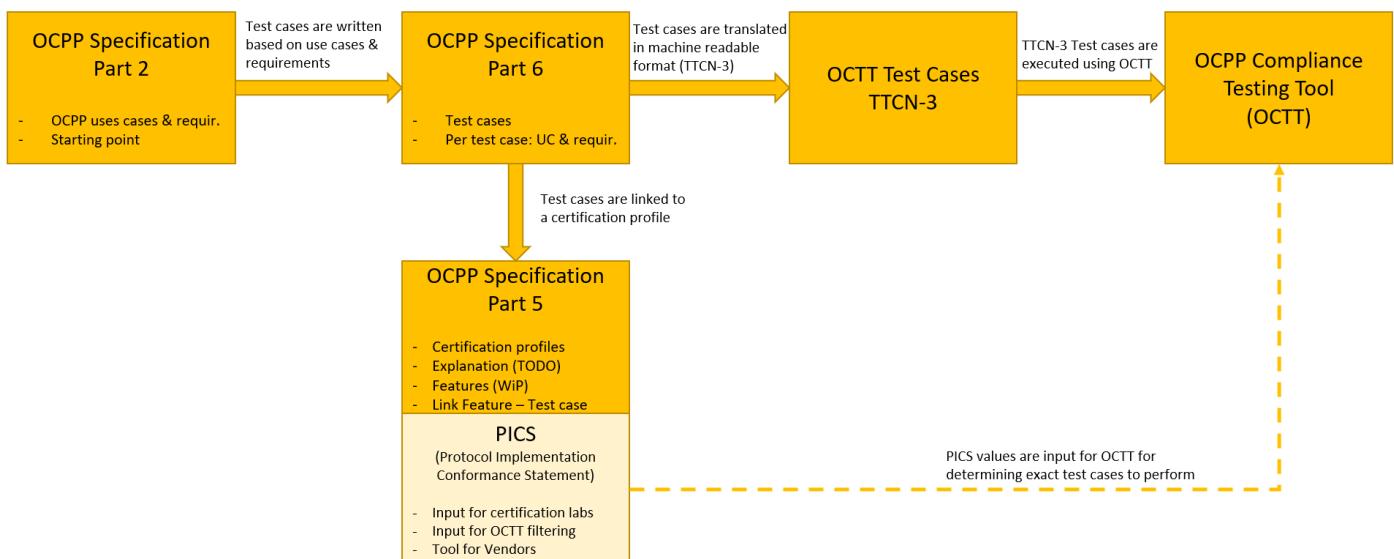


Figure 1. Link between different OCPP Documents in OCPP Certification Program

2. Certification profiles

The OCPP protocol has been designed to support a wide variety of charging stations ranging from simple AC home chargers to advanced DC hyperchargers and megawatt chargers. It will be obvious that these charging stations and associated CSMSs will have very different capabilities. As a result it does not make sense to require every vendor to certify for the full OCPP functionality, when only subset is needed for the specific application.

The OCPP certification is built around certification profiles that describe a set of supported functions. A full OCPP certification comprises all certification profiles, but it is possible to get certified for a subset, since not all OCPP functionality may be needed for some vendors.

The OCPP "Core" profile must always be present. It contains the basic OCPP functionality. On top of that other profiles can be added to the certification. These profiles are independent of each other, the only exception being the "ISO 15118 support" profile, which requires a number of "Advanced security" and "Smart charging" test cases to be implemented.

The following table lists the certification profiles and an overview of the functionality per profile:

Table 1. Certification profiles

Certification Profile	Description
Core	Basic Authentication TLS - server-side certificate Update Charging Station Password for HTTP Basic Authentication Security Event Notification Booting a Charging Station Configuring a Charging Station Resetting a Charging Station / EVSE Authorization incl. GroupId Stop Transaction with a Master Pass Local start transaction - Cable plugin first & Authorization first Start / Stop transaction options Disconnect cable on EV-side Check Transaction status Remote start / stop transaction Remote unlock Connector Remote Trigger Change Availability - Charging Station / EVSE / Connector Clock-aligned Meter & Sampled Meter Values Install CA certificates Retrieve certificates from Charging Station Delete a certificate from a Charging Station AdditionalRootCertificateCheck Retrieve Log Information Get / Clear Customer Information Secure Firmware Update Store / Clear Authorization Data in Authorization Cache Authorization through authorization cache
Advanced Security	TLS - Client-side certificate Update Charging Station Certificate Upgrade Charging Station Security Profile
Local Authorization List Management	Authorization through local authorization list Send Local Authorization List Get Local List Version
Smart Charging	Set charging profile Remote start transaction with charging profile Get Composite Schedule Get Charging Profile Clear Charging Profile

Certification Profile	Description
Advanced Device Management	Get Monitoring report Set Monitoring Base Set Variable Monitoring Set Monitoring Level Get Custom Report Clear / Remove Monitoring Event Notification
Advanced User Interface	Set Display Message Get Display Message Clear a Display Message Show EV Driver Running / Final Total Cost During / After Charging Show EV Driver-specific Tariff Information Update Tariff Information During Transaction Configure Fallback Tariff Information & Total Cost Message
Reservation	Reserve a specific EVSE Reserve an unspecified EVSE Reserve a connector with a specific type Reservations using GroupIdToken Cancel reservation of an EVSE
ISO 15118 support	<p><i>ISO 15118 Certificate Management:</i></p> Update Charging Station Certificate (Contract) Certificate Installation / Update EV Get Certificate Status Install V2G / MO CA certificates Retrieve V2G / MO certificates from Charging Station Delete a certificate from a Charging Station <p><i>ISO 15118 EIM / PnC Authorization:</i></p> Authorization using External Identification Means Authorization using Contract Certificates <p><i>ISO 15118 Smart Charging:</i></p> Set charging profile Remote start transaction with charging profile Get Composite Schedule Get Charging Profile Clear Charging Profile Renegotiating a Charging Schedule ISO 15118 signed meter values

3. Features

The concept of certification profiles is not enough to cover the variety in OCPP implementations. The OCPP specification contains many optional features, often in the form of optional message fields or configuration variables, that a vendor can use to support advanced functions. Whereas the certification profiles determine *which* OCPP functionality is implemented, the features describe *how much* of a certain functionality in a profile has been implemented.

The OCTT test tool uses the features to determine which test cases have to be executed for a charging station or CSMS. For example, the set of [TxStartPoints](#) that a charging station supports, has a big impact on the execution of certain test cases. The behavior of a charging station that starts a transaction based on a successful authorization is different from a charging station that starts a transaction as soon as a cable is connected. Similarly, a CSMS that only controls DC fast chargers will not need functionality to unlock a cable at the charging station. For such a CSMS the vendor may decide to not implement the feature [Support for unlocking connector](#).

In most cases a feature corresponds the existence of a configuration variable or its value.

3.1. Optional feature list for charging station

The following table lists the optional features. These are features that are not mandatory to implement within a certification profile. Where applicable the associated configuration variable is mentioned in parenthesis.

Table 2. Optional features for charging stations

Id	Feature	Charging Station
Core		
C-01	Support for offline authorization of transactions	Optional. Supporting this feature depends on whether at least one of the following is supported; - Certification Profile: Local Authorization List Management - C-02: Support for allowing offline authorization for unknown ids - C-49: Authorization Cache (AuthCacheEnabled)
C-02	Support for allowing Offline Authorization for Unknown Ids (OfflineTxForUnknownIdEnabled)	Optional
C-03	Support for maximizing energy for invalid ids (MaxEnergyOnInvalidId)	Optional
C-04	Support to limit StatusNotifications (MinimumStatusDuration)	Optional
C-05	Support for changing WebSocketPingInterval (WebSocketPingInterval)	Optional
C-06	Authorization status after cable disconnected on EV side (StopTxOnEVSideDisconnect)	{ list } at least one
C-06.1	Support for maintaining authorization when cable disconnected on EV side	Optional
C-06.2	Support for not maintaining authorization when cable disconnected on EV side	Optional
C-07	Support for using a Master Pass for charging stations with UI (MasterPassGroupId)	Optional
C-08	Support for using a Master Pass for charging stations without UI (MasterPassGroupId)	Optional
C-09	Supported Transaction Start points (TxStartPoint)	{ list } at least one
C-09.1	Start transaction options - EVConnected	Optional
C-09.2	Start transaction options - Authorized	Optional
C-09.3	Start transaction options - DataSigned	Optional
C-09.4	Start transaction options - PowerPathClosed	Optional
C-09.5	Start transaction options - EnergyTransfer	Optional
C-09.6	Start transaction options - ParkingBayOccupancy	Optional
C-10	Supported Transaction Stop points (TxStopPoint)	{ list } at least one
C-10.1	Stop transaction options - EVConnected	Optional
C-10.2	Stop transaction options - Authorized	Optional
C-10.3	Stop transaction options - PowerPathClosed	Optional
C-10.4	Stop transaction options - EnergyTransfer	Optional
C-10.5	Stop transaction options - ParkingBayOccupancy	Optional
C-12	Unlocking of connector when cable disconnected on EV side (UnlockOnEVSideDisconnect)	{ list } at least one
C-12.1	Support for unlocking connector when cable disconnected on EV side	Optional
C-12.2	Support for not unlocking when cable disconnected on EV side	Optional
C-13	Support for Reset per EVSE (AllowReset)	Optional
C-14	Support for retrieving / deleting CustomerInformation - CustomerIdentifier	Optional
C-20	Allowing New Sessions Pending a FirmwareUpdate (AllowNewSessionsPendingFirmwareUpdate)	Optional
C-21	Support for queuing all or only Transaction related messages until they are delivered to the CSMS (QueueAllMessages)	Optional

Id	Feature	Charging Station
<i>Time related settings</i>		
C-23	Supported time sources (TimeSource)	{ list } at least Heartbeat
C-25	Support for setting a TimeOffset (TimeOffset)	Optional
C-26	Support for setting the TimeZone (TimeZone)	Optional
C-28	Toggle sending clock aligned meter values when a transaction is ongoing / Idle (AlignedDataSendDuringIdle)	Optional
C-29	TriggerMessage	{ list } 1 or more (Heartbeat MUST be supported) (Security related triggers are separate.)
C-29.1	Trigger message - MeterValues	Optional
C-29.2	Trigger message - TransactionEvent	Optional
C-29.3	Trigger message - LogStatusNotification	Optional
C-29.4	Trigger message - FirmwareStatusNotification	Optional
C-29.5	Trigger message - StatusNotification	Optional
C-29.6	Trigger message - BootNotification	Optional
<i>Authorization options for local start</i>		
C-30	Authorization - using RFID ISO14443	Optional
C-31	Authorization - using RFID ISO15693	Optional
C-32	Authorization - using KeyCode	Optional
C-33	Authorization - using locally generated id	Optional
C-34	Authorization - MacAddress	Optional
C-35	Authorization - NoAuthorization	Optional
<i>Authorization options for remote start (mandatory to support at least one)</i>		
C-36	Authorization - using RFID ISO14443	Optional
C-37	Authorization - using RFID ISO15693	Optional
C-38	Authorization - using centrally, in the CSMS (or other server) generated id	Optional
C-39	Authorization - NoAuthorization	Optional
C-40	Supported MeterValue Measurands	
C-40.1	SampledTxStartedMeasurands	{ list of supported } at least one
C-40.2	SampledTxUpdatedMeasurands	{ list of supported } at least one
C-40.3	SampledTxEndedMeasurands	{ list of supported } at least one
C-40.4	AlignedDataMeasurands	{ list of supported } at least one
C-40.5	AlignedDataTxEndedMeasurands	{ list of supported } at least one
C-41	Supported Cipher Suites	{ list of cipher suites } → at least one of TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 OR TLS_RSA_WITH_AES_128_GCM_SHA256 TLS_RSA_WITH_AES_256_GCM_SHA384
C-42	Signed Metervales (SampledDataSignReadings)	Optional
C-43	Install Firmware with ongoing transaction(s) (AllowNewSessionsPendingFirmwareUpdate)	Optional
C-47	Support for falling back to default OCPP reconnection mechanism when NetworkConnection profile connection has failed	Optional
C-48	Authorization of remote start (AuthorizeRemoteStart)	{ list } at least one
C-48.1	Option for authorization in case of a remote start	Optional
C-48.2	Option for no authorization in case of a remote start	Optional

Id	Feature	Charging Station
C-58	Option for disabling remote authorization (DisableRemoteAuthorization)	Optional
C-49	Authorization Cache (AuthCacheEnabled)	Optional
C-59	Option for disabling remote authorization for cached invalid idTokens (AuthCacheDisablePostAuthorize)	Optional
C-51	Configurable TxStartPoint	Optional
C-52	Configurable TxStopPoint	Optional
C-53	Support for lifetime cached token (AuthCacheLifeTime)	Optional
C-54	Supported policies for replacing cached entries (AuthCachePolicy)	{ list } at least one that is / are supported.
C-56	Support for providing the SummaryInventory	Optional
C-57	Support for cancelling ongoing log file upload	Optional
C-60	Support for cancelling ongoing firmware update	Optional

Advanced Security

AS-2	Additional root certificate check mechanism implemented (AdditionalRootCertificateCheck)	Optional
AS-3	Update Charging Station Certificate - CertificateSignedRequest Timeout (CertSigningWaitMinimum,CertSigningRepeatTimes)	Optional

3.2. Optional feature list for CSMS

The features of a CSMS are not determined by configuration variables. Features in the list below are allowed not to be supported by a CSMS.

Table 3. Optional features for CSMS

Id	Feature	CSMS
Core		
C-11	Support for unlocking connector for charging station with detachable cable (UnlockConnector message).	Optional
C-13	Support for Reset per EVSE	Optional
C-14	Support for retrieving / deleting CustomerInformation - CustomerIdentifier	Optional
C-15	Support for scheduled firmware updates	Optional
C-16	Support for checking the TransactionStatus	Optional
C-17	Support for retrieving the ConfigurationInventory	Optional
C-29	TriggerMessage	{ list } 0 or more (Security related triggers are separate.)
C-29.1	Trigger message - MeterValues	Optional
C-29.2	Trigger message - TransactionEvent	Optional
C-29.3	Trigger message - LogStatusNotification	Optional
C-29.4	Trigger message - FirmwareStatusNotification	Optional
C-29.5	Trigger message - StatusNotification	Optional
<i>Authorization options for local start</i>		
C-30	Authorization - using RFID ISO14443	Required
C-31	Authorization - using RFID ISO15693	Required
C-32	Authorization - using KeyCode	Optional
C-33	Authorization - using locally generated id	Optional
C-34	Authorization - MacAddress	Optional
C-35	Authorization - NoAuthorization	Required
<i>Authorization options for remote start (<u>mandatory</u> to support at least one)</i>		
C-36	Authorization - using RFID ISO14443	Required
C-37	Authorization - using RFID ISO15693	Required
C-38	Authorization - using centrally, in the CSMS (or other server) generated id	Optional

Id	Feature	CSMS
C-39	Authorization - NoAuthorization	Optional
C-44	Support for sending a BootNotification Pending before Accepting	Optional
C-45	Support for Multiple elements GetVariablesRequest	Optional
C-46	Support for Multiple elements SetVariablesRequest	Optional
C-50	GetBaseReport - FullInventory	{ list } at least one
C-50.1	GetBaseReport - FullInventory - During onboarding	Optional
C-50.2	GetBaseReport - FullInventory - Manual trigger	Optional
Advanced Security		
	No optional features for this profile	

4. List of test cases

4.1. Introduction

This table lists the test cases that are part of the OCPP Certification program. For each of the test cases, the columns "Conf. Test for Charging Station" and "Conf. Test for CSMS" indicate whether the test case is mandatory or not within a Certification Profile. The abbreviations have the following meaning:

- M = Mandatory . This means that IF you implement the certification profile this test case belongs to, you MUST successfully pass this test case.
- C = Conditional. This means that IF you meet a condition, you MUST pass this test case. Most conditions refer to the optional features that are listed in the [Features](#).

4.2. Test Cases Core

					Related features	
OCTT Id	OCPP Compliance Testing Tool scenario	Conf. Test for Charging Station	Conf. test for CSMS	Condition / remark	Feature no.	Feature
	Basic Authentication					
TC_A_01	Valid username/password combination	M	M			
TC_A_02	Username does not equal ChargingStationId		M			
TC_A_03	Invalid password		M			
	Update Charging Station Password for HTTP Basic Authentication					
TC_A_09	Accepted	M	M			
TC_A_10	Rejected	M	M			
	TLS - server-side certificate					
TC_A_04	Valid certificate	M	M			
TC_A_05	Invalid certificate	M				
TC_A_06	TLS version too low	M	M			
	Upgrade Charging Station Security Profile					
TC_A_19	Accepted	M	M			
TC_A_20	No valid CSMSRootCertificate installed	C		If the last CSMSRootCertificate can be removed.	AQ-1	Can the last CSMSRootCertificate be removed?
TC_A_22	Downgrade security profile - Rejected	M				
	Cold Boot Charging Station					

					Related features	
TC_B_01	Accepted	M	M			
TC_B_02	Pending	M	C	CSMS: If Pending mechanism is implemented	C-44	BootNotification Pending
TC_B_03	Rejected	M				
TC_B_30	Pending/Rejected - SecurityError	M	C	For CSMS: if CSMS can be configured to first respond to a BootNotificationRequest with status Pending or Rejected	C-44 or NOT AQ-6	BootNotification Pending or Does the CSMS reject unknown Charging Stations during websocket connection setup?
TC_B_31	Pending/Rejected - TriggerMessage		C	For CSMS: if CSMS can be configured to first respond to a BootNotificationRequest with status Pending or Rejected	C-44 or NOT AQ-6	BootNotification Pending or Does the CSMS reject unknown Charging Stations during websocket connection setup?
Status change during offline period						
TC_B_51	> Offline Threshold	M				
TC_B_52	< Offline Threshold	M				
Get Variables						
TC_B_06	single value	M	M			
TC_B_07	multiple values	M	C	If the CSMS supports multiple elements in a GetVariablesRequest	C-45	multiple values elements GetVariablesRequest
TC_B_32	Unknown component	M				
TC_B_33	Unknown variable	M				
TC_B_34	Not supported attribute type	M				
TC_B_08	limit to maximum number of values	C		If the Charging Station supports BytesPerMessageGetVariables	ORS-5	BytesPerMessageGetVariables
Set Variables						
TC_B_09	single value	M	M			
TC_B_10	multiple values	M	C	If the CSMS supports multiple elements in a SetVariablesRequest	C-46	multiple values elements SetVariablesRequest
TC_B_35	Unknown component	M				
TC_B_36	Unknown variable	M				
TC_B_37	Not supported attribute type	M				
TC_B_11	invalidly formatted values	M				
TC_B_39	Read-only	M				
Get Base Report						
TC_B_12	ConfigurationInventory	M	C		C-17	ConfigurationInventory

					Related features	
TC_B_13	FullInventory	M	C		C-50.2	GetBaseReport - FullInventory - Manual trigger
TC_B_15	Not Supported base report	C		For CS: If reportBase SummaryInventory is not supported. This is the case when Certification Profile Advanced Device Management is not supported.	Not C-56	
TC_B_53	Test mandatory DM variables via FullInventory	M				
	Reset Charging Station					
TC_B_20	Without ongoing transaction - OnIdle	M	M			
TC_B_21	With Ongoing Transaction - OnIdle	M	M			
TC_B_22	With Ongoing Transaction - Immediate	M	M			
TC_B_23	Unavailable persists reset	M				
TC_B_41	With multiple ongoing transactions - OnIdle	C		For CS: if no. of EVSEs > 1	HFS-8 > 1	
	Reset EVSE					
TC_B_25	Without ongoing transaction	C	C		C-13	Reset per EVSE
TC_B_26	With Ongoing Transaction - OnIdle	C	C		C-13	Reset per EVSE
TC_B_27	With Ongoing Transaction - Immediate	C	C		C-13	Reset per EVSE
TC_B_28	Not Supported	C		For CS: Charging Station does not support resetting an individual EVSE	NOT C-13	Reset per EVSE
TC_B_29	With ongoing transaction - Not Supported	C		For CS: Charging Station does not support resetting an individual EVSE	NOT C-13	Reset per EVSE
	Set new NetworkConnectionProfile					
TC_B_42	Accepted		M			
TC_B_43	Rejected	M				
TC_B_44	Failed		M			
	Migrate to new ConnectionProfile					
TC_B_45	Success - Same CSMS Root	M		For CS: at least two configuration slots for networkConnectionProfiles must be supported		
TC_B_46	Fallback mechanism - Same CSMS Root	M		For CS: at least two configuration slots for networkConnectionProfiles must be supported		
TC_B_47	Fallback after NetworkProfileConnectionAttempts per NetworkConfigurationPriority failed - New CSMS Root - New CSMS	C		For CS: at least two configuration slots for networkConnectionProfiles must be supported	AS-2 and C-47	Additional Root Certificate check mechanism implemented & Reconnect after NetworkProfileConnectionAttempts

					Related features	
TC_B_49	Fallback after NetworkProfileConnectionAttempts per NetworkConfigurationPriority failed - Same CSMS Root	C		For CS: at least two configuration slots for networkConnectionProfiles must be supported	C-47	Reconnect after NetworkProfileConnectionAttempts
TC_B_50	Success - New CSMS Root - New CSMS	C		For CS: at least two configuration slots for networkConnectionProfiles must be supported	AS-2	Additional Root Certificate check
	Network Reconnection					
TC_B_57	After connection loss	M				
	Local start transaction					
TC_C_02	Authorization Invalid/Unknown	C	M	Charging Station: - The Charging Station supports at least one of the following local start authorization options C-30, C-31, C-32, C35 - The Charging Station does NOT have a cable lock that prevents the EV driver to connect the EV and EVSE before authorization.	(C-30 or C-31 or C-32 or C-35) and NOT AQ-2	Local Authorization - using RFID ISO14443 / RFID ISO15693 / KeyCode / NoAuthorization and Does the Charging Station have a cable lock, which prevents the EV driver to connect the EV and EVSE before authorization?
TC_C_06	Authorization Blocked	C	M	For CS: - The Charging Station supports at least one of the following local start authorization options: C-30, C-31, C-32 - The Charging Station does NOT have a cable lock, which prevents the EV driver to connect the EV and EVSE before authorization.	NOT AQ-2 and (C-30 or C-31 or C-32)	Local Authorization - using RFID ISO14443 / RFID ISO15693 / KeyCode
TC_C_07	Authorization Expired	C	M	For CS: - The Charging Station supports at least one of the following local start authorization options : C-30, C-31, C-32 - The Charging Station does NOT have a cable lock, which prevents the EV driver to connect the EV and EVSE before authorization.	NOT AQ-2 and (C-30 or C-31 or C-32)	Local Authorization - using RFID ISO14443 / RFID ISO15693 / KeyCode
TC_E_38	EV not ready	C	M	For CS: not supporting start transaction options EnergyTransfer	NOT C-09.5 and NOT Product Subtype "Mode 1/2-only Charging Station"	Start transaction options - EnergyTransfer
TC_C_56	Authorization Unknown	C		Charging Station: - The Charging Station supports at least one of the following local start authorization options: C-30, C-31, C-32	C-30 or C-31 or C-32	Local Authorization - using RFID ISO14443 / RFID ISO15693 / KeyCode

					Related features	
TC_C_05	Authorization invalid - Cable lock	C		For CS: - The Charging Station has a cable lock, which prevents the EV driver to connect the EV and EVSE before authorization. - The Charging Station supports at least one of the following local start authorization options C-30, C-31, C-32, C35 - The Charging Station does NOT have the following configuration: TxStartPoint ReadOnly AND value Authorized is NOT set.	(C-30 or C-31 or C-32 or C-35) and AQ-2	Local Authorization - using RFID ISO14443 / RFID ISO15693 / KeyCode / NoAuthorization
	Local Stop Transaction					
TC_C_04	Different idToken	C		Charging Station: - The Charging Station supports at least one of the following local start authorization options: C-30, C-31, C-32	C-30 or C-31 or C-32	Local Authorization - using RFID ISO14443 / RFID ISO15693 / KeyCode
TC_E_06	Accepted	C		The Charging Station supports E07 Transaction locally stopped by IdToken with at least one of the following local start authorization options: C-30, C-31, C-32, C35	C-30 or C-31 or C-32 or C35	Local Authorization - using RFID ISO14443 / RFID ISO15693 / KeyCode / NoAuthorization
	Authorization by GroupId					
TC_C_39	Success	C	M	For CS: the Charging Station supports at least one of the following local start authorization options: C-30, C-31, C-32	C-30 or C-31 or C-32	Local Authorization - using RFID ISO14443 / RFID ISO15693 / KeyCode
TC_C_45	Master pass - Not able to start transaction + groupId	C		For CS: the Charging Station supports at least one of the following local start authorization options: C-30, C-31, C-32 and Master Pass	(C-30 or C-31 or C-32) AND (C-07 OR C-08)	Local Authorization - using RFID ISO14443 / RFID ISO15693 / KeyCode and Master Pass
TC_C_42	Not stopped by GroupId	C		For CS: the Charging Station supports at least one of the following local start authorization options: C-30, C-31, C-32	C-30 or C-31 or C-32	Local Authorization - using RFID ISO14443 / RFID ISO15693 / KeyCode
	Offline Authorization					
TC_C_26	Unknown Id	C		If the feature Unknown Offline Authorization is supported AND the Charging Station supports at least one of the following local start authorization options: C-30, C-31, C-32	C-02 and (C-30 or C-31 or C-32)	Unknown Offline Authorization
	Stop Transaction with a Master Pass					
TC_C_47	With UI - All transactions	C	M	CS: If the feature Master Pass with UI is supported AND the Charging Station supports at least one of the following local start authorization options: C-30, C-31, C-32	C-07 and (C-30 or C-31 or C-32)	Master Pass - With UI

					Related features	
TC_C_48	With UI - With UI - Specific transactions	C	M	CS: If the feature Master Pass with UI is supported AND the Charging Station supports at least one of the following local start authorization options: C-30, C-31, C-32	C-07 and (C-30 or C-31 or C-32)	Master Pass - With UI
TC_C_49	Without UI	C	M	CS: If the feature Master Pass with UI is supported AND the Charging Station supports at least one of the following local start authorization options: C-30, C-31, C-32	C-08 and (C-30 or C-31 or C-32)	Master Pass - Without UI
	Store Authorization Data in the Authorization Cache					
TC_C_32	Persistent over reboot	C		If the Charging Station has an authorization cache, then it must support this use case AND the Charging Station supports at least one of the following local start authorization options: C-30, C-31, C-32	C-49 and (C-30 or C-31 or C-32)	Authorization Cache
TC_C_33	Update on AuthorizeResponse	C		If the Charging Station has an authorization cache, then it must support this use case AND the Charging Station supports at least one of the following local start authorization options: C-30, C-31, C-32	C-49 and (C-30 or C-31 or C-32)	Authorization Cache
TC_C_34	Update on TransactionResponse	C		If the Charging Station has an authorization cache, then it must support this use case AND the Charging Station supports at least one of the following local start authorization options: C-30, C-31, C-32	C-49 and (C-30 or C-31 or C-32)	Authorization Cache
TC_C_36	AuthCacheCtrlr.LocalPreAuthorize = false	C		If the Charging Station has an authorization cache and AuthCacheEnabled is implemented AND the Charging Station supports at least one of the following local start authorization options: C-30, C-31, C-32	C-49 and (C-30 or C-31 or C-32)	Authorization Cache
TC_C_46	AuthCacheLifeTime	C		If the Charging Station has an authorization cache, then it must support this use case AND the Charging Station supports at least one of the following local start authorization options: C-30, C-31, C-32	C-53 and (C-30 or C-31 or C-32)	AuthCacheLifeTime
	Clear Authorization Data in Authorization Cache					
TC_C_37	Accepted	C	M	If the Charging Station has an authorization cache, then it must support this use case AND the Charging Station supports at least one of the following local start authorization options: C-30, C-31, C-32	C-49 and (C-30 or C-31 or C-32)	Authorization Cache

					Related features	
TC_C_38	Rejected	C	M	If the Charging Station has an authorization cache and AuthCacheEnabled is implemented AND the Charging Station supports at least one of the following local start authorization options: C-30, C-31, C-32	C-49 and (C-30 or C-31 or C-32)	Authorization Cache
	Authorization by GroupId					
TC_C_41	Success with Authorization Cache	C		For CS: - The Charging Station supports at least one of the following local start authorization options: C-30, C-31, C-32 - If the Charging Station has an authorization cache.	C-49 and (C-30 or C-31 or C-32)	Authorization Cache
TC_C_44	Invalid status with Authorization Cache	C		For CS: - The Charging Station supports at least one of the following local start authorization options: C-30, C-31, C-32 - If the Charging Station has an authorization cache.	C-49 and (C-30 or C-31 or C-32)	Authorization Cache
	Authorization through authorization cache					
TC_C_08	Accepted	C	M	If the Charging Station has an authorization cache AND the Charging Station supports at least one of the following local start authorization options: C-30, C-31, C-32	C-49 and (C-30 or C-31 or C-32)	Authorization Cache
TC_C_09	Invalid & Not Accepted	C		If the Charging Station has an authorization cache AND the Charging Station supports at least one of the following local start authorization options: C-30, C-31, C-32	C-49 and (C-30 or C-31 or C-32)	Authorization Cache
TC_C_12	Invalid & Accepted	C		If the Charging Station has an authorization cache AND the Charging Station supports at least one of the following local start authorization options: C-30, C-31, C-32	C-49 and (C-30 or C-31 or C-32)	Authorization Cache
TC_C_10	Blocked	C		If the Charging Station has an authorization cache AND the Charging Station supports at least one of the following local start authorization options: C-30, C-31, C-32	C-49 and (C-30 or C-31 or C-32)	Authorization Cache
TC_C_11	Expired	C		If the Charging Station has an authorization cache AND the Charging Station supports at least one of the following local start authorization options: C-30, C-31, C-32	C-49 and (C-30 or C-31 or C-32)	Authorization Cache

					Related features	
TC_C_13	Accepted but cable not connected yet.	C		If the Charging Station has an authorization cache AND the Charging Station supports at least one of the following local start authorization options: C-30, C-31, C-32	C-49 and (C-30 or C-31 or C-32)	Authorization Cache
TC_C_15	StopTxOnInvalidId = false, MaxEnergyOnInvalidId > 0	C		If the Charging Station has an authorization cache AND the Charging Station supports at least one of the following local start authorization options: C-30, C-31, C-32 If MaxEnergyOnInvalidId is implemented.	C-49 and C-03 and (C-30 or C-31 or C-32)	Authorization Cache & MaxEnergyOnInvalidId
TC_C_16	StopTxOnInvalidId = true	C		If the Charging Station has an authorization cache AND the Charging Station supports at least one of the following local start authorization options: C-30, C-31, C-32	C-49 and (C-30 or C-31 or C-32)	Authorization Cache
TC_C_17	StopTxOnInvalidId = false	C		If the Charging Station has an authorization cache AND the Charging Station supports at least one of the following local start authorization options: C-30, C-31, C-32	C-49 and (C-30 or C-31 or C-32)	Authorization Cache
TC_C_18	StopTxOnInvalidId = true, MaxEnergyOnInvalidId > 0	C		If the Charging Station has an authorization cache AND the Charging Station supports at least one of the following local start authorization options: C-30, C-31, C-32 If MaxEnergyOnInvalidId is implemented.	C-49 and C-03 and (C-30 or C-31 or C-32)	Authorization Cache & MaxEnergyOnInvalidId
TC_C_20	Invalid		M			
TC_C_57	AuthCacheDisablePostAuthorize	C		If the Charging Station supports the option for disabling remote authorization for cached invalid idTokens AND has an authorization cache AND the Charging Station supports at least one of the following local start authorization options: C-30, C-31, C-32	C-59 and C-49 and (C-30 or C-31 or C-32)	
Local start transaction - Cable plugin first						
TC_E_03	Success	C	M	Applicable if one or more of the local start authorization options is implemented.	NOT AQ-2 and (C-30 - C-35 or ISO 15118 support)	Authorization options for local start
Local start transaction - Authorization first						
TC_E_04	Success	C	M	Applicable if one or more of the local start authorization options is implemented.	C-30 - C-35 or ISO 15118 support	Authorization options for local start
TC_E_05	Cable plugin timeout	C		Applicable if one or more of the local start authorization options is implemented.	C-30 - C-35 or ISO 15118 support	Authorization options for local start
TC_E_52	DisableRemoteAuthorization	C		If the Charging Station supports the option for disabling remote authorization	C-58	

					Related features	
	Start transaction options					
TC_E_09	EVConnected	C	M	TxStartPoint can either be ReadOnly with a subset of the values or have a valueList of supported values, that contains a subset. This testcase is applicable if the value EVConnected is a supported value. And it should be possible to not set ParkingBayOccupancy.	C-09.1 and (C-51 or NOT C-09.6)	
TC_E_10	Authorized - Local	C	M	TxStartPoint can either be ReadOnly with a subset of the values or have a valueList of supported values, that contains a subset. This testcase is applicable if the value Authorized is a supported value. - If one or more of the local start authorization options is implemented.	C-09.2 and (C-30 - C-35 or ISO 15118 support)	Supported Transaction Start Points & Authorization options for local start & Authorization - eMAID
TC_E_13	Authorized - Remote	C		TxStartPoint can either be ReadOnly with a subset of the values or have a valueList of supported values, that contains a subset. This testcase is applicable if the value Authorized is a supported value.	C-09.2	Supported Transaction Start points
TC_E_11	DataSigned	C	M	CS: TxStartPoint can either be ReadOnly with a subset of the values or have a valueList of supported values, that contains a subset. This testcase is applicable if the value DataSigned is a supported value. And it should be possible to not set ParkingBayOccupancy and EVConnected and Authorized. CSMS: Must at least be able to receive a signed MeterValue. It does not need to be able to read it.	C-09.3 and (C-51 or NOT (C-09.1 or C-09.2 or C-09.6))	Supported Transaction Start points
TC_E_01	PowerPathClosed	C	M	TxStartPoint can either be ReadOnly with a subset of the values or have a valueList of supported values, that contains a subset. This testcase is applicable if the value PowerPathClosed is a supported value. And it should be possible to not set ParkingBayOccupancy and EVConnected and Authorized and DataSigned.	C-09.4 and (C-51 or NOT (C-09.1 or C-09.2 or C-09.3 or C-09.6))	Supported Transaction Start points

					Related features	
TC_E_02	EnergyTransfer	C	M	TxStartPoint can either be ReadOnly with a subset of the values or have a valueList of supported values, that contains a subset. This testcase is applicable if the value EnergyTransfer is a supported value. And it should be possible to not set ParkingBayOccupancy and EVConnected and Authorized and DataSigned and PowerPathClosed	C-09.5 and (C-51 or NOT (C-09.1 or C-09.2 or C-09.3 or C-09.4 or C-09.6))	Supported Transaction Start points
TC_E_12	ParkingBayOccupied	C	M	TxStartPoint can either be ReadOnly with a subset of the values or have a valueList of supported values, that contains a subset. This testcase is applicable if the value ParkingBayOccupied is a supported value.	C-09.6	Supported Transaction Start points
Stop transaction options						
TC_E_14	EVDisconnected - Charging Station side	C	M	TxStopPoint can either be ReadOnly with a subset of the values or have a valueList of supported values, that contains a subset. This testcase is applicable if the value EVConnected is a supported value. And it should be possible to not set EnergyTransfer and PowerPathClosed and Authorized. Charging Station does NOT have a fixed cable.	HFS-1 and C-10.1 and (C-52 or NOT (C-10.2 or C-10.3 or C-10.4))	Supported Transaction Stop points
TC_E_20	EVDisconnected - EV side (able to charge IEC 61851-1 EV)	C	M	TxStopPoint can either be ReadOnly with a subset of the values or have a valueList of supported values, that contains a subset. This testcase is applicable if the value EVConnected is a supported value. And it should be possible to not set EnergyTransfer and PowerPathClosed AND The Charging Station does NOT have following configuration combination; StopTxOnEVSideDisconnect mutability ReadOnly with value true AND TxStopPoint mutability is ReadOnly and contains Authorized	C-10.1 and (C-52 or NOT (C-10.2 or C-10.3 or C-10.4)) AND NOT C-06.1) AND (AQ-9 OR Product Subtype "Mode 1/2-only Charging Station")	Supported Transaction Stop points

					Related features	
TC_E_54	EVDisconnected - EV side (not able to charge IEC 61851-1 EV)	C		<p>TxStopPoint can either be ReadOnly with a subset of the values or have a valueList of supported values, that contains a subset. This testcase is applicable if the value EVConnected is a supported value. And it should be possible to not set EnergyTransfer and PowerPathClosed AND</p> <p>The Charging Station does NOT have following configuration combination;</p> <p>StopTxOnEVSideDisconnect mutability ReadOnly with value true AND TxStopPoint mutability is <i>ReadOnly</i> and contains <i>Authorized</i></p>	C-10.1 and (C-52 or NOT (C-10.2 or C-10.3 or C-10.4)) AND (HFS-4 OR ISO15118 support) AND NOT Product Subtype "Mode 1/2-only Charging Station"	Supported Transaction Stop points
TC_E_15	StopAuthorized - Local	C	M	<p>TxStopPoint can either be ReadOnly with a subset of the values or have a valueList of supported values, that contains a subset. This testcase is applicable if the value Authorized is a supported value.</p> <p>The Charging Station supports E07 Transaction locally stopped by IdToken with at least one of the following local start authorization options: C-30, C-31, C-32, C35</p>	C-10.2 and (C-30 or C-31 or C-32 or C35)	Supported Transaction Stop Points & Local Authorization - using RFID ISO14443 / RFID ISO15693 / KeyCode / NoAuthorization
TC_E_21	StopAuthorized - Remote	C	M	<p>TxStopPoint can either be ReadOnly with a subset of the values or have a valueList of supported values, that contains a subset. This testcase is applicable if the value Authorized is a supported value.</p>	C-10.2	Supported Transaction Stop points
TC_E_16	Deauthorized - Invalid idToken	C	M	<p>TxStopPoint can either be ReadOnly with a subset of the values or have a valueList of supported values, that contains a subset. This testcase is applicable if the value Authorized or PowerPathClosed is a supported value.</p> <p>Charging Station: If one or more of the local start authorization options is implemented. AND either a cache, local authorization list or UnknownIdtag (C15) is implemented.</p>	(C-10.2 or C-10.3) and (C-30 - C-35 or ISO 15118 support) and C-01	Supported Transaction Stop Points & Local Authorization options for local start & Authorization - eMAID
TC_E_17	Deauthorized - EV side disconnect	C	M	<ul style="list-style-type: none"> - TxStopPoint can either be ReadOnly with a subset of the values or have a valueList of supported values, that contains a subset. This testcase is applicable if the value Authorized or PowerPathClosed is a supported value. - StopTxOnEVSideDisconnect needs to ReadWrite or ReadOnly with value true 	(C-10.2 or C-10.3) and C-06.2 and AQ-9	Supported Transaction Stop points

					Related features	
TC_E_39	Deauthorized - timeout	C	M	TxStopPoint can either be ReadOnly with a subset of the values or have a valueList of supported values, that contains a subset. This testcase is applicable if the value Authorized is a supported value.	C-10.2	Supported Transaction Stop points
TC_E_07	PowerPathClosed - Local stop	C	M	TxStopPoint can either be ReadOnly with a subset of the values or have a valueList of supported values, that contains a subset. This testcase is applicable if the value PowerPathClosed is a supported value. And it should be possible to not set Authorized. The Charging Station supports E07 Transaction locally stopped by IdToken with at least one of the following local start authorization options: C-30, C-31, C-32, C35	C-10.3 and (C-52 or NOT C-10.2) and (C-30 or C-31 or C-32 or C35)	Supported Transaction Stop Points & Local Authorization - using RFID ISO14443 / RFID ISO15693 / KeyCode / NoAuthorization
TC_E_35	PowerPathClosed - Remote stop	C		TxStopPoint can either be ReadOnly with a subset of the values or have a valueList of supported values, that contains a subset. This testcase is applicable if the value PowerPathClosed is a supported value. And it should be possible to not set Authorized.	C-10.3 and (C-52 or NOT C-10.2)	Supported Transaction Stop points
TC_E_37	PowerPathClosed - EV side disconnect	C		TxStopPoint can either be ReadOnly with a subset of the values or have a valueList of supported values, that contains a subset. This testcase is applicable if the value PowerPathClosed is a supported value. And it should be possible to not set EnergyTransfer and EVConnected.	C-10.3 and (C-52 or NOT (C-10.1 or C-10.4)) AND (AQ-9 OR Product Subtype "Mode 1/2-only Charging Station")	Supported Transaction Stop points
TC_E_08	EnergyTransfer stopped - StopAuthorized	C	M	TxStopPoint can either be ReadOnly with a subset of the values or have a valueList of supported values, that contains a subset. This testcase is applicable if the value EnergyTransfer is a supported value. And it should be possible to not set PowerPathClosed and Authorized.	C-10.4 and (C-52 or NOT (C-10.2 or C-10.3))	Supported Transaction Stop points
TC_E_22	EnergyTransfer stopped - SuspendedEV	C	M	TxStopPoint can either be ReadOnly with a subset of the values or have a valueList of supported values, that contains a subset. This testcase is applicable if the value EnergyTransfer is a supported value.	C-10.4	Supported Transaction Stop points

					Related features	
TC_E_19	ParkingBayUnoccupied	C	M	TxStopPoint can either be ReadOnly with a subset of the values or have a valueList of supported values, that contains a subset. This testcase is applicable if the value ParkingBayOccupancy is a supported value. And it should be possible to not set EnergyTransfer and Authorized and PowerPathClosed and EVConnected.	C-10.5 and (C-52 or NOT (C-10.1 or C-10.2 or C-10.3 or C-10.4))	Supported Transaction Stop points
	Disconnect cable on EV-side					
TC_E_24	Deauthorize transaction - UnlockOnEVSideDisconnect is true	C		The Charging Station does NOT have a permanently attached cable. UnlockOnEVSideDisconnect can be set to true StopTxOnEVSideDisconnect can be set to true	HFS-1 and C-06.2 and C-12.1 and AQ-9 and NOT Product Subtype "Mode 1/2-only Charging Station"	Support for not maintaining authorization when cable disconnected on EV side & Support for unlocking connector when cable disconnected on EV side
TC_E_25	Deauthorize transaction - UnlockOnEVSideDisconnect is false	C		UnlockOnEVSideDisconnect can be set to false StopTxOnEVSideDisconnect can be set to true	C-06.2 and C-12.2 and (AQ-9 OR Product Subtype "Mode 1/2-only Charging Station")	Support for not maintaining authorization when cable disconnected on EV side & Support for not unlocking connector when cable disconnected on EV side
TC_E_26	Suspend transaction	C	M	TxStopPoint can either be ReadOnly with a subset of the values or have a valueList of supported values, that contains a subset. This testcase is applicable if the value ParkingBayOccupancy or Authorized is a supported value. And it should be possible to not set EnergyTransfer and PowerPathClosed and EVConnected. UnlockOnEVSideDisconnect can be set to false StopTxOnEVSideDisconnect can be set to false	(C-10.2 or C-10.5) and (C-52 or NOT (C-10.1 or C-10.3 or C-10.4)) and C-06.1 and C-12.2 and AQ-9 and NOT Product Subtype "Mode 1/2-only Charging Station"	
TC_E_27	Suspend transaction - Fixed cable connection timeout	C		TxStopPoint can either be ReadOnly with a subset of the values or have a valueList of supported values, that contains a subset. This testcase is applicable if the value ParkingBayOccupancy or Authorized is a supported value. And it should be possible to not set EnergyTransfer and PowerPathClosed and EVConnected. The Charging Station has a permanently attached cable at the Charging Station side. UnlockOnEVSideDisconnect can be set to false StopTxOnEVSideDisconnect can be set to false	(C-10.2 or C-10.5) and (C-52 or NOT (C-10.1 or C-10.3 or C-10.4)) and C-06.1 and C-12.2 and HFS-2 and AQ-9 and NOT Product Subtype "Mode 1/2-only Charging Station"	

					Related features	
	Retry sending transaction message when failed					
TC_E_41	Max retry count reached	M				
TC_E_42	Success before reaching the max retry count	M				
TC_E_50	Max retry count reached - CallError	M				
TC_E_51	Success before reaching the max retry count - CallError	M				
	Offline Behaviour					
TC_E_40	Connection loss during transaction	M				
TC_E_43	Transaction during offline period	C		Charging Station: If one or more of the local start authorization options is implemented.	C-01 and (C-30 - C-35 or ISO 15118 support)	Offline transaction support & Local Authorization options for local start
TC_E_44	Stop transaction during offline period	C		Charging Station: If one or more of the local start authorization options is implemented.	C-01 and (C-30 - C-35 or ISO 15118 support)	Offline transaction support & Local Authorization options for local start & Authorization - eMAID
TC_E_45	Stop transaction during offline period - Same GroupId	C		For CS: the Charging Station supports at least one of the following local start authorization options: C-30, C-31, C-32 and Local Authorization List or Authorization Cache	(C-30 or C-31 or C-32) AND (Local Authorization List Management or C-49)	Local Authorization - using RFID ISO14443 / RFID ISO15693 / KeyCode and Local Authorization List or Authorization Cache
	Check Transaction status					
TC_E_28	TransactionId unknown	M				
TC_E_29	Transaction with id ongoing - with message in queue	M	C		C-16	Check TransactionStatus
TC_E_30	Transaction with id ongoing - without message in queue	M	C		C-16	Check TransactionStatus
TC_E_31	Transaction with id ended - with message in queue	M	C		C-16	Check TransactionStatus
TC_E_32	Transaction with id ended - without message in queue	M				
TC_E_33	Without transactionId - with message in queue	M	C		C-16	Check TransactionStatus
TC_E_34	Without transactionId - without message in queue	M	C		C-16	Check TransactionStatus
	Reset Sequence Number					
TC_E_53	CSMS accepting seqNo = 0 at start of transaction		M			
	Remote start transaction					
TC_F_01	Cable plugin first	C	M	If the Charging Station does not have a cable lock.	NOT AQ-2 and (C-36 -(or) C-39)	Authorization options for remote start

					Related features	
TC_F_02	Remote start first - AuthorizeRemoteStart is true	C	M	If AuthorizeRemoteStart can be set to true	C-48.1 and (C-36 -(or) C-39)	Authorization options for remote start
TC_F_03	Remote start first - AuthorizeRemoteStart is false	C	M	If AuthorizeRemoteStart can be set to false	C-48.2 and (C-36 -(or) C-39)	Authorization options for remote start
TC_F_04	Remote start first - Cable plugin timeout	M	M			
	Remote stop transaction					
TC_F_08	Success	M				
TC_F_09	Rejected	M				
	Remote unlock Connector					
TC_F_05	With ongoing transaction	C		If the Charging Station has a detachable cable.	HFS-1 and NOT Product Subtype "Mode 1/2-only Charging Station"	
TC_F_06	Without ongoing transaction - Accepted	HFS-1 and NOT Product Subtype "Mode 1/2-only Charging Station"	C-11	If the Charging Station has a detachable cable. If the CSMS support the Unlocking connector for Charging Station with detachable cable (UnlockConnector) feature.	See column 3/4	
TC_F_07	Without ongoing transaction - No cable connected	C		If the Charging Station has a detachable cable.	HFS-1 and NOT Product Subtype "Mode 1/2-only Charging Station"	
TC_F_10	Without ongoing transaction - UnknownConnector	C		If the Charging Station has a detachable cable.	HFS-1 and NOT Product Subtype "Mode 1/2-only Charging Station"	
	Trigger message					
TC_F_11	MeterValues - Specific EVSE	C	C	If the SUT supports TriggerMessage for requestedMessage MeterValues for a specific EVSE.	C-29.1	TriggerMessage
TC_F_12	MeterValues - All EVSE	C	C	If the SUT supports TriggerMessage for requestedMessage MeterValues for all EVSE.	C-29.1	TriggerMessage
TC_F_13	TransactionEvent - Specific EVSE	C	C	If the SUT supports TriggerMessage for requestedMessage TransactionEvent for a specific EVSE.	C-29.2	TriggerMessage

					Related features	
TC_F_14	TransactionEvent - All EVSE	C	C	If the SUT supports TriggerMessage for requestedMessage TransactionEvent for all EVSE.	C-29.2	TriggerMessage
TC_F_15	LogStatusNotification - Idle	C	C	If the SUT supports TriggerMessage for requestedMessage LogStatusNotification.	C-29.3	TriggerMessage
TC_F_16	LogStatusNotification - Uploading	C		If the Charging Station supports TriggerMessage for requestedMessage LogStatusNotification.	C-29.3	TriggerMessage
TC_F_17	FirmwareStatusNotification - Specific EVSE not relevant	C		If the Charging Station supports TriggerMessage for requestedMessage FirmwareStatusNotification.	C-29.4	TriggerMessage
TC_F_18	FirmwareStatusNotification - Idle	C	C	If the SUT supports TriggerMessage for requestedMessage FirmwareStatusNotification.	C-29.4	TriggerMessage
TC_F_19	FirmwareStatusNotification - Downloading	C		If the Charging Station supports TriggerMessage for requestedMessage FirmwareStatusNotification.	C-29.4	TriggerMessage
TC_F_20	Heartbeat	M	M			
TC_F_23	StatusNotification - Specific EVSE - Available	C	C	If the SUT supports TriggerMessage for requestedMessage StatusNotification for a specific EVSE.	C-29.5	TriggerMessage
TC_F_24	StatusNotification - Specific EVSE - Occupied	C	C	If the SUT supports TriggerMessage for requestedMessage StatusNotification for a specific EVSE.	C-29.5	TriggerMessage
TC_F_26	BootNotification - Rejected	C		If the Charging Station supports TriggerMessage for requestedMessage BootNotification.	C-29.6	TriggerMessage
TC_F_27	NotImplemented	C	M	For CS: can only be done when SignCombinedCertificate is notimplemented	NOT ISO-3	
	Connector status Notification			Charging Station: This can either be implemented with the StatusNotification or NotifyEvent message. CSMS: Both StatusNotification and NotifyEvent must be supported.		
TC_G_01	Available to Occupied	M				
TC_G_02	Occupied to Available	M				
TC_G_20	Lock Failure		M			

					Related features	
	Change Availability EVSE			Charging Station: This can either be implemented with the StatusNotification or NotifyEvent message. CSMS: Both StatusNotification and NotifyEvent must be supported.		
TC_G_03	Operative to inoperative	M	M			
TC_G_09	Operative to operative	M				
TC_G_04	Inoperative to operative	M	M			
TC_G_10	Inoperative to inoperative	M				
TC_G_11	With ongoing transaction	M	M			
TC_G_18	state persists across reboot	M				
	Change Availability Charging Station			Charging Station: This can either be implemented with the StatusNotification or NotifyEvent message. CSMS: Both StatusNotification and NotifyEvent must be supported.		
TC_G_05	Operative to inoperative	M	M			
TC_G_12	Operative to operative	M				
TC_G_06	Inoperative to operative	M	M			
TC_G_13	Inoperative to inoperative	M				
TC_G_21	state persists across reboot	M				
TC_G_14	With ongoing transaction	M	M			
	Change Availability Connector			Charging Station: This can either be implemented with the StatusNotification or NotifyEvent message. CSMS: Both StatusNotification and NotifyEvent must be supported.		
TC_G_07	Operative to inoperative	M	M			
TC_G_15	Operative to operative	M				
TC_G_08	Inoperative to operative	M	M			
TC_G_16	Inoperative to inoperative	M				
TC_G_17	With ongoing transaction	M	M			
TC_G_19	state persists across reboot	M				

					Related features	
	Clock-aligned Meter Values			Charging Station can choose which measurands are supported (At least one). This can either be implemented with the MeterValues or NotifyEvent message.		
TC_J_01	No transaction ongoing	M	M		C-40	Supported MeterValue Measurands
TC_J_02	Transaction ongoing	M	M		C-40	Supported MeterValue Measurands
TC_J_03	EventType Ended	M	M		C-40	Supported MeterValue Measurands
TC_J_04	Signed	C	M	Charging Station: If signed MeterValues is implemented CSMS: Must at least be able to receive a signed MeterValue. It does not need to be able to read it.	C-40 and C-42	Supported MeterValue Measurands & Signed Metervalues
TC_J_06	No Meter Values during transaction	C		If AlignedDataSendDuringIdle is supported.	C-28	AlignedDataSendDuringIdle
	Sampled Meter Values			Charging Station can choose which measurands are supported (At least one).		
TC_J_07	EventType Started - EVSE known	M	M		C-40	Supported MeterValue Measurands
TC_J_08	Context Transaction.Begin - EVSE not known	C	M		C-40 and NOT AQ-8 AND (C-09.2 OR C-09.6)	Supported MeterValue Measurands & possibility to enforce EVSE being known.
TC_J_09	EventType Updated	M	M		C-40	Supported MeterValue Measurands
TC_J_10	EventType Ended	M	M		C-40	Supported MeterValue Measurands
TC_J_11	Signed	C	M	Charging Station: If signed MeterValues is implemented CSMS: Must at least be able to receive a signed MeterValue. It does not need to be able to read it.	C-42	Supported MeterValue Measurands & Signed Metervalues
	Remote start transaction with charging profile					
TC_K_38	Ignore chargingProfile	C		The Charging Station does NOT support Smart Charging.	NOT Smart Charging	
	Secure Firmware Update					
TC_L_01	Installation successful	M	M			
TC_L_02	InstallScheduled	M	C		C-15	Scheduled firmware updates
TC_L_03	DownloadScheduled	M	C		C-15	Scheduled firmware updates
TC_L_04	RevokedCertificate		M			
TC_L_05	InvalidCertificate	M	M			
TC_L_06	InvalidSignature	M	M			
TC_L_07	DownloadFailed	M	M			

					Related features	
TC_L_08	InstallVerificationFailed or InstallationFailed	M	M			
TC_L_09	InstallationFailed		M			
TC_L_10	AcceptedCanceled	C	M	The Charging Station supports cancelling an ongoing firmware update	C-60	
TC_L_11	Unable to cancel	C	M	The Charging Station does NOT supports cancelling an ongoing firmware update	NOT C-60	
TC_L_18	Missing firmware signing certificate and signature	M				
TC_L_12	Unable to download/install firmware with ongoing transaction - AllowNewSessionsPendingFirmwareUpdate is true	C		AllowNewSessionsPendingFirmwareUpdate is implemented. The Charging Station is unable to download AND install firmware while there is an ongoing transaction.	C-20 and NOT C-43 and NOT AQ-7 and HFS-8 > 1	AllowNewSessionsPendingFirmwareUpdate
TC_L_13	Unable to download/install firmware with ongoing transaction - AllowNewSessionsPendingFirmwareUpdate is false	C	M	AllowNewSessionsPendingFirmwareUpdate is implemented. The Charging Station is unable to download AND install firmware while there is an ongoing transaction.	NOT C-43 and NOT AQ-7	
TC_L_14	Unable to install firmware with ongoing transaction - AllowNewSessionsPendingFirmwareUpdate is true	C		AllowNewSessionsPendingFirmwareUpdate is implemented. The Charging Station is unable to install firmware while there is an ongoing transaction	C-20 and NOT C-43 and AQ-7 and HFS-8 > 1	AllowNewSessionsPendingFirmwareUpdate
TC_L_15	Unable to install firmware with ongoing transaction - AllowNewSessionsPendingFirmwareUpdate is false	C		AllowNewSessionsPendingFirmwareUpdate is implemented. The Charging Station is unable to install firmware while there is an ongoing transaction	NOT C-43 and AQ-7	
TC_L_16	Able to update firmware with ongoing transaction	C		If the Charging Station supports Install Firmware with ongoing transaction(s)	C-43	Install Firmware with ongoing transaction(s)
Retrieve certificates from Charging Station						
TC_M_12	CSMSRootCertificate	M				
TC_M_13	ManufacturerRootCertificate	M	M			
TC_M_17	CSMSRootCertificate & ManufacturerRootCertificate	M				
TC_M_18	All certificateTypes	M	M			
TC_M_19	No matching certificate found	M	M			
Delete a certificate from a Charging Station						
TC_M_20	Success	M	M			
TC_M_21	Failed		M			

					Related features	
TC_M_22	No matching certificate found	M				
TC_M_23	Unable to delete the Charging Station Certificate	M				
	Install CA certificate					
TC_M_01	CSMSRootCertificate	M	M			
TC_M_02	ManufacturerRootCertificate	M	M			
TC_M_05	Failed		M			
TC_M_07	Rejected - Certificate invalid	M				
TC_M_09	AdditionalRootCertificateCheck - Rejected	C		If the Charging Station supports AdditionalRootCertificateCheck with value true	AS-2	Additional Root Certificate check mechanism implemented
TC_M_30	AdditionalRootCertificateCheck - Reconnect using new CSMS Root - Success	C		If the Charging Station supports AdditionalRootCertificateCheck with value true	AS-2	Additional Root Certificate check mechanism implemented
TC_M_31	AdditionalRootCertificateCheck - Reconnect using new CSMS Root - Fallback mechanism	C		If the Charging Station supports AdditionalRootCertificateCheck with value true	AS-2	Additional Root Certificate check mechanism implemented
	Retrieve Log Information					
TC_N_25	Diagnostics Log - Success	M	M			
TC_N_34	Rejected		M			
TC_N_26	Diagnostics Log - Upload failed	M				
TC_N_35	Security Log - Success	M	M			
TC_N_36	Second Request	C	M	If the Charging Station is able to cancel an ongoing log file upload.	C-57	
	Get Customer Information					
TC_N_27	Accepted + data	C	M	For CS: The Charging Station needs to support Local Authorization and either the Local Authorization List or Authorization Cache. IdToken is used as customer information.	(C-30 or C-31 or C-34) and (Local Authorization List Management or C-49)	
TC_N_28	Accepted + no data	C	M	For CS: The Charging Station needs to support Local Authorization and either the Local Authorization List or Authorization Cache. IdToken is used as customer information.	(C-30 or C-31 or C-34) and (Local Authorization List Management or C-49)	
TC_N_29	Not Accepted		M			
	Clear Customer Information					
TC_N_30	Clear and report + data	C	M	For CS: The Charging Station needs to support Local Authorization or the Authorization Cache. IdToken is used as customer information.	(C-30 or C-31 or C-34) and (Local Authorization List Management or C-49)	

					Related features	
TC_N_31	Clear and report + no data	C	M	For CS: The Charging Station needs to support Local Authorization or the Authorization Cache. IdToken is used as customer information.	(C-30 or C-31 or C-34) and (Local Authorization List Management or C-49)	
TC_N_32	Clear and no report	M	M			
TC_N_62	Clear and report - customerIdentifier	C	C	Support for retrieving / deleting CustomerInformation - CustomerIdentifier	C-14	
	Data Transfer to the Charging Station					
TC_P_01	Rejected / Unknown VendorId / Unknown MessageId	M		Charging Station must be able to Reject the message.		
	Data Transfer to the CSMS					
TC_P_02	Rejected / Unknown VendorId / Unknown MessageId		M	CSMS must be able to Reject the message.		
	CustomData					
TC_P_03	Receive custom data	M	M			

4.3. Test Cases Advanced Security

					Related features	
OCTT Id	OCPP Compliance Testing Tool scenario	Conf. Test for Charging Station	Conf. test for CSMS	Condition / remark	Feature no.	Feature
	TLS - Client-side certificate					
TC_A_07	valid certificate	M	M			
TC_A_08	Invalid certificate		M			
	Update Charging Station Certificate by request of CSMS					
TC_A_11	Success - Charging Station Certificate	M	M			
TC_A_14	Invalid certificate	M	M			
TC_A_15	SignCertificateRequest Rejected	M				
TC_A_23	CertificateSignedRequest Timeout	C		If the Charging Station supports CertificateSignedRequest Timeout	AS-3	
	Upgrade Charging Station Security Profile					
TC_A_21	No valid ChargingStationCertificate installed	C		If the last ChargingStationCertificate can be removed (Via other means than OCPP).	AQ-1	

4.4. Test Cases Local Authorization List Management

List will become available in a later version of this document.

4.5. Test Cases Smart Charging

List will become available in a later version of this document.

4.6. Test Cases Advanced Device Management

List will become available in a later version of this document.

4.7. Test Cases Reservation

List will become available in a later version of this document.

4.8. Test Cases Advanced User Interface

List will become available in a later version of this document.

4.9. Test Cases ISO 15118 Support

List will become available in a later version of this document.

5. OCPP 2.0.1 Mandatory Controller components per profile

Controller components contain variables that describe the supported features of a Charging Station and influence its behavior. In OCPP 2.0.1 we have configuration variables that are required or optional, but these are contained by controller components. Functionalities cannot be tested without the accompanying controller component, so for certification the following controller components are mandatory:

Certification Profile	Description
Core	OCPPCommCtrlr TxCtrlr DeviceDataCtrlr ClockCtrlr SecurityCtrlr SampledDataCtrlr AlignedDataCtrlr AuthCtrlr
Advanced Security	SecurityCtrlr (already part of Core)
Smart Charging	SmartChargingCtrlr
ISO 15118 Support	ISO15118Ctrlr SmartChargingCtrlr
Advanced Diagnostics	MonitoringCtrlr
Local Authorization List Management	LocalAuthListCtrlr
Advanced UI	TariffCostCtrlr DisplayMessageCtrlr
Reservation	ReservationCtrlr

6. Appendix A: additional questions for the Protocol Implementation Conformance Statement

To perform the certification testing, the test lab need some additional information (for the test selection). This concerns the following questions:

6.1. Questions for Charging Stations

Id	Additional questions for lab testing
AQ-1	Can the last CSMSRootCertificate be removed?
AQ-2	Does the Charging Station have a cable lock, which prevents the EV driver to connect the EV and EVSE before authorization?
AQ-3	Can the last ChargingStationCertificate be removed (via other means than OCPP)?
AQ-4	Is there at least one unsupported NumberOfPhases?
AQ-5	Does the Charging Station have at least one hardWired monitor? If yes, which hardWired monitor should be used for the certification test
AQ-6	Does the Charging Station have a pre-configured monitor? If yes, which pre-configured monitor should be used for the certification test
AQ-7	Is your Charging Station able to download firmware while there is an ongoing transaction?
AQ-8	Does your Charging Station enforce a selection of EVSE (by design) prior to authorization?
AQ-9	Does your Charging Station support charging an EV using IEC 61851-1 (Mode 3)?

6.2. Questions for CSMSs

Id	Additional questions for lab testing
AQ-1	Can your CSMS be configured to first respond to a BootNotificationRequest with status Pending or Rejected?
AQ-2	Is a FullInventory requested during onboarding / booting test cases?
AQ-6	Does the CSMS reject unknown Charging Stations during websocket connection setup?

7. Appendix B: Hardware feature set

The table below gives an overview of the hardware feature set IDs that are used for determining whether test cases are needed / applicable for certification.

Table 4. Hardware features

Id	Hardware Feature
HFS-1	Charging Station has a detachable cable
HFS-2	Charging Station has a fixed cable
HFS-3	Charging Station has AC support
HFS-4	Charging Station has DC support
HFS-5	Charging Station has 1 phase support
HFS-6	Charging Station has 2 phase support
HFS-7	Charging Station has 3 phase support
HFS-8	No. EVSEs of Charging Station

8. Appendix C: Features vs. OCPP use cases

The table below gives an overview of the use cases / configuration variables that the features are applicable for / referring to in the OCPP 2.0.1 Specification Part 2.

Table 5. Optional features vs. related use cases

Id	Feature	Related use cases
Core		
C-01	Support for offline authorization of transactions	C15, C10, C11, C12
C-02	Support for allowing Offline Authorization for Unknown Ids (OfflineTxForUnknownIdEnabled)	C15
C-03	Support for maximizing energy for invalid ids (MaxEnergyOnInvalidId)	C15, E05
C-04	Support to limit StatusNotifications (MinimumStatusDuration)	Configuration Variable for G01
C-05	Support for changing WebSocketPingInterval (WebSocketPingInterval)	Configuration Variable related (B05, B06)
C-06	Authorization status after cable disconnected on EV side (StopTxOnEVSideDisconnect)	
C-06.1	Support for maintaining authorization when cable disconnected on EV side	E10
C-06.2	Support for not maintaining authorization when cable disconnected on EV side	E09
C-07	Support for using a Master Pass for charging stations with UI (MasterPassGroupId)	C16
C-08	Support for using a Master Pass for charging stations without UI (MasterPassGroupId)	C16
C-09	Supported Transaction Start points (TxStartPoint)	E01
C-09.1	Start transaction options - EVConnected	E01-S2
C-09.2	Start transaction options - Authorized	E01-S3
C-09.3	Start transaction options - DataSigned	E01-S4
C-09.4	Start transaction options - PowerPathClosed	E01-S5
C-09.5	Start transaction options - EnergyTransfer	E01-S6
C-09.6	Start transaction options - ParkingBayOccupancy	E01-S1
C-10	Supported Transaction Stop points (TxStopPoint)	E06
C-10.1	Stop transaction options - EVConnected	E06-S2
C-10.2	Stop transaction options - Authorized	E06-S3
C-10.3	Stop transaction options - PowerPathClosed	E06-S5
C-10.4	Stop transaction options - EnergyTransfer	E06-S6
C-10.5	Stop transaction options - ParkingBayOccupancy	E06-S1
C-12	Unlocking of connector when cable disconnected on EV side (UnlockOnEVSideDisconnect)	E09, E10
C-12.1	Support for unlocking connector when cable disconnected on EV side	E09, E10
C-12.2	Support for not unlocking when cable disconnected on EV side	E09, E10
C-13	Support for Reset per EVSE (AllowReset)	B11, B12
C-14	Support for retrieving / deleting CustomerInformation - CustomerIdentifier	N09, N10
C-20	Allowing New Sessions Pending a FirmwareUpdate (AllowNewSessionsPendingFirmwareUpdate)	Configuration Variable for L01
C-21	Support for queuing all or only Transaction related messages until they are delivered to the CSMS (QueueAllMessages)	Optional
<i>Time related settings</i>		
C-23	Supported time sources (TimeSource)	
C-25	Support for setting a TimeOffset (TimeOffset)	Configuration Variable (B05, B06)
C-26	Support for setting the TimeZone (TimeZone)	Configuration Variable (B05, B06)
C-28	Toggle sending clock aligned meter values when a transaction is ongoing / Idle (AlignedDataSendDuringIdle)	Configuration Variable for J01
C-29	TriggerMessage	F06

Id	Feature	Related use cases
C-29.1	Trigger message - MeterValues	F06
C-29.2	Trigger message - TransactionEvent	F06
C-29.3	Trigger message - LogStatusNotification	F06
C-29.4	Trigger message - FirmwareStatusNotification	F06
C-29.5	Trigger message - StatusNotification	F06
C-29.6	Trigger message - BootNotification	F06
<i>Authorization options for local start</i>		
C-30	Authorization - using RFID ISO14443	C01
C-31	Authorization - using RFID ISO15693	C01
C-32	Authorization - using KeyCode	C04
C-33	Authorization - using locally generated id	C06
C-34	Authorization - MacAddress	C06
C-35	Authorization - NoAuthorization	C02
<i>Authorization options for remote start (<u>mandatory</u> to support at least one)</i>		
C-36	Authorization - using RFID ISO14443	C01
C-37	Authorization - using RFID ISO15693	C01
C-38	Authorization - using centrally, in the CSMS (or other server) generated id	C05
C-39	Authorization - NoAuthorization	C02
C-40	Supported MeterValue Measurands (SampledDataTx{Started,Updated,Ended}Measurands, AlignedDataMeasurands)	J01, J02
C-41	Supported Cipher Suites	See requirement A00.FR.318, A00.FR.319, A00.FR.421, A00.FR.422
C-42	Signed Metervalues (SampledDataSignReadings)	J01, J02
C-43	Install Firmware with ongoing transaction(s) (AllowNewSessionsPendingFirmwareUpdate)	Configuration Variable for L01
C-47	Support for falling back to default OCPP reconnection mechanism when NetworkConnection profile connection has failed	B10 (FR.07)
C-48	Authorization of remote start (AuthorizeRemoteStart)	F01, F02
C-48.1	Option for authorization in case of a remote start	F01, F02
C-48.2	Option for no authorization in case of a remote start	F01, F02
C-58	Option for disabling remote authorization (DisableRemoteAuthorization)	Configuration Variable (B05, B06)
C-49	Authorization Cache (AuthCacheEnabled)	C10, C11, C12
C-59	Option for disabling remote authorization for cached invalid idTokens (AuthCacheDisablePostAuthorize)	Configuration Variable for C10, C12
C-51	Configurable TxStartPoint	Configuration Variable for E01
C-52	Configurable TxStopPoint	Configuration Variable for E06
C-53	Support for lifetime cached token (AuthCacheLifeTime)	Configuration Variable for C10
C-54	Supported policies for replacing cached entries (AuthCachePolicy)	Configuration Variable for C10, C11, C12
C-56	Support for providing the SummaryInventory	B07
C-57	Support for cancelling ongoing log file upload	N01 (AcceptedCanceled)
C-60	Support for cancelling ongoing firmware update	L01, L02 (AcceptedCanceled)
Advanced Security		
AS-2	Additional root certificate check mechanism implemented (AdditionalRootCertificateCheck)	Configuration Variable for M05
AS-3	Update Charging Station Certificate - CertificateSignedRequest Timeout (CertSigningWaitMinimum,CertSigningRepeatTimes)	Configuration Variable for A02, A03



OCPP 2.0.1

Part 6 - Test Cases

Core & Advanced Security, FINAL, 2023-06-30

Table of Contents

1. Introduction	2
1.1. About this document	2
1.2. Conventions	2
2. Test Cases Charging Station	3
2.1. General pre conditions & tool validations	3
2.2. A Security	4
2.3. B Provisioning	25
2.4. C Authorization	82
2.5. D Local Authorization List Management	120
2.6. E Transactions	120
2.7. F Remote Control	171
2.8. G Availability	195
2.9. H Reservation	217
2.10. I Tariff and Cost	217
2.11. J MeterValues	217
2.12. K SmartCharging	230
2.13. L Firmware Management	232
2.14. M ISO IEC 15118 CertificateManagement	268
2.15. N Diagnostics	283
2.16. O Display Message	295
2.17. P DataTransfer	295
2.18. Reusable states	297
2.19. Memory states	317
3. Test Cases Charging Station Management System	324
3.1. General pre/post conditions & tool validations	324
3.2. A Security	325
3.3. B Provisioning	342
3.4. C Authorization	363
3.5. D Local Authorization List Management	375
3.6. E Transactions	375
3.7. F Remote Control	404
3.8. G Availability	420
3.9. H Reservation	430
3.10. I Tariff and Cost	430
3.11. J MeterValues	430
3.12. K SmartCharging	440
3.13. L Firmware Management	440
3.14. M ISO IEC 15118 CertificateManagement	459
3.15. N Diagnostics	467
3.16. O Display Message	478
3.17. P DataTransfer	478
3.18. Reusable states	480

Copyright © 2010 - 2023 Open Charge Alliance. All rights reserved.

This document is made available under the **Creative Commons Attribution-NoDerivatives 4.0 International Public License** (<https://creativecommons.org/licenses/by-nd/4.0/legalcode>).

1. Introduction

1.1. About this document

This document is created to describe a set of valid test cases for OCPP 2.0.1. These test cases can be executed using the OCPP Compliance Testing Tool (OCTT) for OCPP 2.0.1. The scenarios in the tool are described in detail including the expected behaviour of the System Under Test (SUT). This document is divided in chapters, each describing an OCPP functional block as can be found in the official OCPP specification. These are:

- A. Security
- B. Provisioning
- C. Authorization
- D. Local Authorization List Management
- E. Transactions
- F. Remote Control
- G. Availability
- H. Reservation
- I. Tariff and Cost
- J. Meter Values
- K. Smart Charging
- L. Firmware Management
- M. ISO 15118 Certificate Management
- N. Diagnostics
- O. Display Message
- P. Data Transfer

The scenarios in this document are also part of the OCA certification process of OCPP. Please refer to OCPP 2.0.1 Part 5 - Certification Profiles for more information about the relation between certification profiles and the test scenarios in this document.

1.2. Conventions

The following conventions / rules apply to all test cases, unless explicitly mentioned otherwise. These will not be mentioned separately at every test case.

- The OCPP specification is always leading.
- This document does not specify which tests need to be passed for certification, this will be specified in a separate document.
- All messages shall comply with the OCPP 2.0.1 schemas from the OCPP specification.
- The messages are to be sent as mentioned in the scenario details.
- Validations will be mentioned and grouped per step.
- Messages, datatypes and configuration variables will convey to the following formatting rules:
 - Datatypes, messages and configuration variables are displayed bold.
 - Values are displayed italic.

2. Test Cases Charging Station

2.1. General pre conditions & tool validations

General conditions/validations are overruled by testcase specific conditions/validations, unless specifically stated otherwise.

General pre conditions:

- Charging Station is Accepted by the CSMS
- Charging Station has a stable active connection to the CSMS
- Charging Station connectors are available
- Charging Station is Idle, with no active transactions
- Charging Station is clear of faults
- Charging Station has no charging schedules active
- Charging Station has no active reservations
- The Configuration variable **AuthCtrlr.LocalPreAuthorize** is set to *false*.
- Charging Station has no more OCPP messages to be send in queue
- Charging Station is not busy with transfer of diagnostics
- Charging Station is not busy with download of firmware
- Charging Station is not upgrading firmware
- Charging Station is ready to accept/start a charging session
- Charging Station has no Display message configured
- Charging Station has no active custom monitors

General tool rules/validations:

- TransactionEventRequest messages don't have to be sent in chronological order. However the provided seqNo are sequentially numbered in chronological order. This way the CSMS is able to determine whether all messages of a transaction have been received.
- After connecting/disconnecting the EV and EVSE, the Charging Station SHALL report the new status of its connector and report any queued TransactionEventRequest(s). These message are allowed to be sent in any order.
- If the transaction was authorized with **Reusable State Authorized** remote, then the first TransactionEventRequest sent after receiving a **RequestStartTransactionRequest** message will contain **triggerReason** with value **_RemoteStart** (This will overrule the step specific tool validations) AND will contain **transactionInfo.remoteStartId**
- The first **TransactionEventRequest** of a transaction MUST contain **eventType Started**.
- The first **TransactionEventRequest** sent after connecting the EVSE and EV MUST contain **evse.id** and **evse.connectorId**
- The first **TransactionEventRequest** sent after presenting the idToken MUST contain **idToken** with value <Configured valid idToken fields>
- If the energy transfer was stopped with **Reusable State StopAuthorized** local, then the **_stoppedReason** of the last **TransactionEventRequest** of that transaction with **eventType Ended**, must have value **Local** OR be omitted.
- When validating/comparing time / dateTIme values, the OCTT will in most cases accept a configurable deviation. The certification labs will configure a deviation of 4 seconds.
- Every FirmwareStatusNotificationRequest sent for a firmware update SHALL contain the same requestId as the UpdateFirmwareRequest that started the firmware update.
- The list of ChargingSchedulePeriod elements in a chargingSchedule SHALL be ordered by increasing values of ChargingSchedulePeriod.startPeriod. This means the list is in chronological order.

2.2. A Security

Table 1. Test Case Id: TC_A_01_CS

Test case name	Basic Authentication - Valid username/password combination	
Test case Id	TC_A_01_CS	
Use case Id(s)	A00, B01	
Requirement(s)	A00.FR.202, A00.FR.203, A00.FR.204, A00.FR.205, A00.FR.301, A00.FR.302, A00.FR.304 AND B01.FR.01, B01.FR.05, B01.FR.09	
System under test	Charging Station	
Description	The Charging Station uses Basic authentication to authenticate itself to the CSMS, when using security profile 1 or 2.	
Purpose	To verify whether the Charging Station is able to authenticate itself to the CSMS using Basic Authentication.	
Prerequisite(s)	<ul style="list-style-type: none"> - The charging station supports security profile 1 and/or 2 - The active NetworkConnectionProfile uses either security profile 1 OR 2. 	
Before (Preparations)	<p>Configuration State: SecurityCtrlr.BasicAuthPassword is <Configured basicAuthPassword></p> <p>Memory State: N/a</p> <p>Reusable State(s): N/a</p>	
Main (Test scenario)	Charging Station	CSMS
	1. Execute Reusable State Booted	
Tool validations	<p>* Step 1: The authorization header of the HTPP upgrade request must be formatted as follows: AUTHORIZATION: Basic <Base64 encoded(<ChargingStationId>:<Configured basicAuthPassword>)></p> <ul style="list-style-type: none"> - The ChargingStationId, must equal the ChargingStationId provided at the end of the connection url string of the HTTP request. - BasicAuthPassword must consist of minimum 16 and maximum 40 characters - BasicAuthPassword may only contain alpha-numeric characters and the special characters allowed by identifierString. <p>Post scenario validations: N/a</p>	

Table 2. Test Case Id: TC_A_04_CS

Test case name	TLS - server-side certificate - Valid certificate																	
Test case Id	TC_A_04_CS																	
Use case Id(s)	A00																	
Requirement(s)	A00.FR.309,A00.FR.312,A00.FR.313,A00.FR.319,A00.FR.321,A00.FR.412,A00.FR.422																	
System under test	Charging Station																	
Description	The CSMS uses a server-side certificate to identify itself to the Charging Station, when using security profile 2 or 3.																	
Purpose	To verify whether the Charging Station is able to receive a server certificate provided by the CSMS and setup a secured WebSocket connection.																	
Prerequisite(s)	<ul style="list-style-type: none"> - The charging station supports security profile 2 and/or 3 - The active NetworkConnectionProfile uses either security profile 2 OR 3. 																	
Before (Preparations)	<p>Configuration State: N/a</p> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>Booting</i></p>																	
Main (Test scenario)	<table border="1"> <thead> <tr> <th>Charging Station</th> <th>CSMS</th> </tr> </thead> <tbody> <tr> <td>1. The Charging Station initiates a TLS handshake and sends a Client Hello to the OCTT.</td> <td>2. The OCTT responds with a Server Hello With the <Configured server certificate></td> </tr> <tr> <td>3. The Charging Station performs the following actions: Send client certificate Client Key Exchange Certificate verify Change Cipher Spec Finished</td> <td>4. The OCTT performs the following actions: Change Cipher Spec Finished</td> </tr> <tr> <td>Note(s): - <i>The client certificate is only sent when the Charging Station uses security profile 3.</i></td> <td></td> </tr> <tr> <td>5. The Charging Station sends a HTTP upgrade request to the OCTT</td> <td>6. The OCTT upgrades the connection to a (secured) WebSocket connection.</td> </tr> <tr> <td>Note(s): - <i>The HTTP request only contains a username/password combination when the Charging Station uses security profile 2.</i></td> <td></td> </tr> <tr> <td>7. The Charging Station sends a BootNotificationRequest</td> <td>8. The OCTT responds with a BootNotificationResponse with status Accepted</td> </tr> <tr> <td>9. The Charging Station notifies the CSMS about the current state of all connectors.</td> <td>10. The OCTT responds accordingly.</td> </tr> </tbody> </table>		Charging Station	CSMS	1. The Charging Station initiates a TLS handshake and sends a Client Hello to the OCTT.	2. The OCTT responds with a Server Hello With the <Configured server certificate>	3. The Charging Station performs the following actions: Send client certificate Client Key Exchange Certificate verify Change Cipher Spec Finished	4. The OCTT performs the following actions: Change Cipher Spec Finished	Note(s): - <i>The client certificate is only sent when the Charging Station uses security profile 3.</i>		5. The Charging Station sends a HTTP upgrade request to the OCTT	6. The OCTT upgrades the connection to a (secured) WebSocket connection.	Note(s): - <i>The HTTP request only contains a username/password combination when the Charging Station uses security profile 2.</i>		7. The Charging Station sends a BootNotificationRequest	8. The OCTT responds with a BootNotificationResponse with status Accepted	9. The Charging Station notifies the CSMS about the current state of all connectors.	10. The OCTT responds accordingly.
Charging Station	CSMS																	
1. The Charging Station initiates a TLS handshake and sends a Client Hello to the OCTT.	2. The OCTT responds with a Server Hello With the <Configured server certificate>																	
3. The Charging Station performs the following actions: Send client certificate Client Key Exchange Certificate verify Change Cipher Spec Finished	4. The OCTT performs the following actions: Change Cipher Spec Finished																	
Note(s): - <i>The client certificate is only sent when the Charging Station uses security profile 3.</i>																		
5. The Charging Station sends a HTTP upgrade request to the OCTT	6. The OCTT upgrades the connection to a (secured) WebSocket connection.																	
Note(s): - <i>The HTTP request only contains a username/password combination when the Charging Station uses security profile 2.</i>																		
7. The Charging Station sends a BootNotificationRequest	8. The OCTT responds with a BootNotificationResponse with status Accepted																	
9. The Charging Station notifies the CSMS about the current state of all connectors.	10. The OCTT responds accordingly.																	

Test case name	TLS - server-side certificate - Valid certificate
Tool validations	<p>* Step 2: The OCTT validates the following before sending the server certificate:</p> <ul style="list-style-type: none"> - The Charging Station must use TLS version 1.2 or above <p>At least the following set of cipher suites must be supported:</p> <p>(TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 AND TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384) OR (TLS_RSA_WITH_AES_128_GCM_SHA256 AND TLS_RSA_WITH_AES_256_GCM_SHA384)</p> <p>* Step 9: Message: StatusNotificationRequest - connectorStatus Available Message: NotifyEventRequest - eventData[0].trigger Delta - eventData[0].actualValue "Available" - eventData[0].component.name "Connector" - eventData[0].variable.name "AvailabilityState"</p>
	Post scenario validations: N/a

Table 3. Test Case Id: TC_A_05_CS

Test case name	TLS - server-side certificate - Invalid certificate	
Test case Id	TC_A_05_CS	
Use case Id(s)	A00	
Requirement(s)	A00.FR.309,A00.FR.310,A00.FR.311,A00.FR.412,A00.FR.413,A00.FR.414	
System under test	Charging Station	
Description	The CSMS uses a server-side certificate to identify itself to the Charging Station, when using security profile 2 or 3.	
Purpose	To verify whether the Charging Station is able to terminate the connection when the received server certificate is invalid.	
Prerequisite(s)	<ul style="list-style-type: none"> - The charging station supports security profile 2 and/or 3 - The active NetworkConnectionProfile uses either security profile 2 OR 3. - This testcase can be executed multiple times, using different kinds of invalid certificates: Unknown certificate expired certificate certificate with commonName that does not equal the FQDN of the CSMS.	
Before (Preparations)	Configuration State: OCPPCommCtrlr.NetworkProfileConnectionAttempts is 2	
	Memory State: N/a	
	Reusable State(s): State is <i>Booting</i>	
Main (Test scenario)	Charging Station	CSMS
	1. The Charging Station initiates a TLS handshake and sends a Client Hello to the OCTT.	2. The OCTT responds with a Server Hello With a <Configured invalid server certificate>
	3. The Charging Station deems the server certificate invalid and terminates the connection.	
	4. The Charging Station initiates a TLS handshake and sends a Client Hello to the OCTT.	5. The OCTT responds with a Server Hello With the <Configured server certificate>

Test case name	TLS - server-side certificate - Invalid certificate	
	<p>6. The Charging Station performs the following actions:</p> <ul style="list-style-type: none"> Send client certificate Client Key Exchange Certificate verify Change Cipher Spec Finished <p>Note(s):</p> <ul style="list-style-type: none"> - <i>The client certificate is only sent when the Charging Station uses security profile 3.</i> 	<p>7. The OCTT performs the following actions:</p> <ul style="list-style-type: none"> Change Cipher Spec Finished
	<p>8. The Charging Station sends a HTTP upgrade request to the OCTT</p> <p>Note(s):</p> <ul style="list-style-type: none"> - <i>The HTTP request only contains a username/password combination when the Charging Station uses security profile 2.</i> 	<p>9. The OCTT upgrades the connection to a (secured) WebSocket connection.</p>
	<p>10. The Charging Station sends a BootNotificationRequest</p>	<p>11. The OCTT responds with a BootNotificationResponse with status Accepted</p>
	<p>12. The Charging Station notifies the CSMS about the current state of all connectors.</p>	<p>13. The OCTT responds accordingly.</p>
	<p>14 The Charging Station sends a SecurityEventNotificationRequest</p>	<p>15 The OCTT responds with a SecurityEventNotificationResponse</p>
Tool validations	<p>* Step 14:</p> <p>Message: SecurityEventNotificationRequest</p> <ul style="list-style-type: none"> - type must be <i>InvalidCsmsCertificate</i> <p>Post scenario validations:</p> <p>N/a</p>	

Table 4. Test Case Id: TC_A_06_CS

Test case name	TLS - server-side certificate - TLS version too low	
Test case Id	TC_A_06_CS	
Use case Id(s)	A00	
Requirement(s)	A00.FR.314,A00.FR.316,A00.FR.416,A00.FR.417,A00.FR.419	
System under test	Charging Station	
Description	The CSMS uses a server-side certificate to identify itself to the Charging Station, when using security profile 2 or 3.	
Purpose	To verify whether the Charging Station is able to terminate the connection when it notices the used TLS version is lower than 1.2.	
Prerequisite(s)	<ul style="list-style-type: none"> - The charging station supports security profile 2 and/or 3 - The active NetworkConnectionProfile uses either security profile 2 OR 3. 	
Before (Preparations)	<p>Configuration State: OCPPCommCtrlr.NetworkProfileConnectionAttempts is 1</p> <p>Memory State: N/a</p> <p>Reusable State(s): N/a</p>	
Main (Test scenario)	<p>Charging Station</p> <p>1. The Charging Station initiates a TLS handshake and sends a Client Hello to the OCTT.</p> <p>3. The Charging Station notices the used TLS version is lower than 1.2 and terminates the connection.</p> <p>4. The Charging Station initiates a TLS handshake and sends a Client Hello to the OCTT.</p>	<p>CSMS</p> <p>2. The OCTT responds with a Server Hello, but uses a TLS version lower than 1.2 With a <Configured server certificate></p> <p>5. The OCTT responds with a Server Hello With the <Configured server certificate></p>

Test case name	TLS - server-side certificate - TLS version too low	
	<p>6. The Charging Station performs the following actions:</p> <ul style="list-style-type: none"> Send client certificate Client Key Exchange Certificate verify Change Cipher Spec Finished <p><u>Note(s):</u></p> <ul style="list-style-type: none"> - <i>The client certificate is only sent when the Charging Station uses security profile 3.</i> 	<p>7. The OCTT performs the following actions:</p> <ul style="list-style-type: none"> Change Cipher Spec Finished
	<p>8. The Charging Station sends a HTTP upgrade request to the OCTT</p> <p><u>Note(s):</u></p> <ul style="list-style-type: none"> - <i>The HTTP request only contains a username/password combination when the Charging Station uses security profile 2.</i> 	<p>9. The OCTT upgrades the connection to a (secured) WebSocket connection.</p>
	<p>10. The Charging Station sends a BootNotificationRequest</p>	<p>11. The OCTT responds with a BootNotificationResponse with status Accepted</p>
	<p>12. The Charging Station notifies the CSMS about the current state of all connectors.</p>	<p>13. The OCTT responds accordingly.</p>
	<p>14 The Charging Station sends a SecurityEventNotificationRequest</p>	<p>15 The OCTT responds with a SecurityEventNotificationResponse</p>
	<p>16 The Charging Station sends a SecurityEventNotificationRequest</p>	<p>17 The OCTT responds with a SecurityEventNotificationResponse</p>
	<p><u>Note(s):</u></p> <ul style="list-style-type: none"> - <i>The order in which the requests of steps 12 and 14 and 16 arrive is not relevant.</i> - <i>Steps 16 and 17 are optional as the Charging Station might not be able to detect that the TLS handshake failed, because of invalid TLS version.</i> 	
Tool validations	<ul style="list-style-type: none"> * Step 14: Message: SecurityEventNotificationRequest - type must be <i>StartupOfTheDevice</i> or <i>ResetOrReboot</i> * Step 16: Message: SecurityEventNotificationRequest - type must be <i>InvalidTLSVersion</i> 	

Table 5. Test Case Id: TC_A_07_CS

Test case name	TLS - Client-side certificate - valid certificate													
Test case Id	TC_A_07_CS													
Use case Id(s)	A00													
Requirement(s)	A00.FR.401,A00.FR.402,A00.FR.415,A00.FR.416,A00.FR.422,A00.FR.502,A00.FR.503,A00.FR.507,A00.FR.508,A00.FR.511													
System under test	Charging Station													
Description	The Charging Station uses a client-side certificate to identify itself to the CSMS, when using security profile 3.													
Purpose	To verify whether the Charging Station is able to provide a valid client certificate and setup a secured WebSocket connection.													
Prerequisite(s)	<ul style="list-style-type: none"> - The charging station supports security profile 3 - The active NetworkConnectionProfile uses security profile 3. 													
Before (Preparations)	<p>Configuration State: N/a</p> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>Booting</i></p>													
Main (Test scenario)	<table border="0"> <tr> <td>Charging Station</td> <td>CSMS</td> </tr> <tr> <td>1. The Charging Station initiates a TLS handshake and sends a Client Hello to the OCTT.</td> <td>2. The OCTT responds with a Server Hello With the <Configured server certificate></td> </tr> <tr> <td>3. The Charging Station performs the following actions: Send client certificate Client Key Exchange Certificate verify Change Cipher Spec Finished</td> <td>4. The OCTT performs the following actions: Change Cipher Spec Finished</td> </tr> <tr> <td>5. The Charging Station sends a HTTP upgrade request to the OCTT</td> <td>6. The OCTT upgrades the connection to a (secured) WebSocket connection.</td> </tr> <tr> <td>7. The Charging Station sends a BootNotificationRequest</td> <td>8. The OCTT responds with a BootNotificationResponse with status Accepted</td> </tr> <tr> <td>9. The Charging Station notifies the CSMS about the current state of all connectors.</td> <td>10. The OCTT responds accordingly.</td> </tr> </table>		Charging Station	CSMS	1. The Charging Station initiates a TLS handshake and sends a Client Hello to the OCTT.	2. The OCTT responds with a Server Hello With the <Configured server certificate>	3. The Charging Station performs the following actions: Send client certificate Client Key Exchange Certificate verify Change Cipher Spec Finished	4. The OCTT performs the following actions: Change Cipher Spec Finished	5. The Charging Station sends a HTTP upgrade request to the OCTT	6. The OCTT upgrades the connection to a (secured) WebSocket connection.	7. The Charging Station sends a BootNotificationRequest	8. The OCTT responds with a BootNotificationResponse with status Accepted	9. The Charging Station notifies the CSMS about the current state of all connectors.	10. The OCTT responds accordingly.
Charging Station	CSMS													
1. The Charging Station initiates a TLS handshake and sends a Client Hello to the OCTT.	2. The OCTT responds with a Server Hello With the <Configured server certificate>													
3. The Charging Station performs the following actions: Send client certificate Client Key Exchange Certificate verify Change Cipher Spec Finished	4. The OCTT performs the following actions: Change Cipher Spec Finished													
5. The Charging Station sends a HTTP upgrade request to the OCTT	6. The OCTT upgrades the connection to a (secured) WebSocket connection.													
7. The Charging Station sends a BootNotificationRequest	8. The OCTT responds with a BootNotificationResponse with status Accepted													
9. The Charging Station notifies the CSMS about the current state of all connectors.	10. The OCTT responds accordingly.													

Test case name	TLS - Client-side certificate - valid certificate
Tool validations	<p>* Step 4:</p> <p>The OCTT validates the following before finishing the TLS handshake:</p> <ul style="list-style-type: none"> - The Charging Station must use TLS version 1.2 or above <p>At least the following set of cipher suites must be supported:</p> <p>(TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 AND TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384) OR (TLS_RSA_WITH_AES_128_GCM_SHA256 AND TLS_RSA_WITH_AES_256_GCM_SHA384)</p> <ul style="list-style-type: none"> - When using RSA or DSA the key must be at least 2048 bits long. <p>and when using elliptic curve cryptography the key must be at least 224 bits long.</p> <ul style="list-style-type: none"> - The received Client side certificate must be transmitted in the X.509 format encoded in Privacy-Enhanced Mail (PEM) format. - The certificate must include a serial number. - The subject field of the certificate must contain a commonName RDN which consists of the unique serial number of the Charging Station. <p><i>NOTE: If one of the above validations fails, the OCTT can still setup the WebSocket connection (if it is able to), but the testcase will FAIL and the OCTT reports why it failed.</i></p> <p>* Step 9:</p> <p>Message: StatusNotificationRequest</p> <ul style="list-style-type: none"> - connectorStatus Available <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none"> - eventData[0].trigger Delta - eventData[0].actualValue "Available" - eventData[0].component.name "Connector" - eventData[0].variable.name "AvailabilityState"
	<p>Post scenario validations:</p> <p>N/a</p>

Table 6. Test Case Id: TC_A_09_CS

Test case name	Update Charging Station Password for HTTP Basic Authentication - Accepted	
Test case Id	TC_A_09_CS	
Use case Id(s)	A01	
Requirement(s)	A01.FR.01, A01.FR.11, A01.FR.12, B01.FR.01	
System under test	Charging Station	
Description	This test case defines how to use the BasicAuthPassword, the password used to authenticate Charging Stations in security profile 1 (Basic Authentication) and security profile 2 (TLS with Basic Authentication)	
Purpose	To verify if the Charging Station is able to accept and store and log the new BasicAuthPassword as described at the OCPP specification.	
Prerequisite(s)	The charging station supports security profile 1 and/or 2	
Before (Preparations)	Configuration State: N/a Memory State: N/a Charging State: N/a	
Main (Test scenario)	Charging Station 2. The Charging Station responds with a SetVariablesResponse 3. The ChargingStation sends a HTTP upgrade request with an Authorization header, containing a username/password combination (with the new <i>BasicAuthPassword</i>). <u>Note(s):</u> - The Authorization header is formatted as follows: <i>AUTHORIZATION: Basic <Base64 encoded(<Configured ChargingStationId>:<NEW BasicAuthPassword>)></i>	CSMS 1. The OCTT sends a SetVariablesRequest with setVariableData[1]: - variable.name = "BasicAuthPassword" - component.name = "SecurityCtrlr" - attributeValue = "<NewPassword>" 4. The OCTT validates the username/password combination AND upgrades the connection to a (secured) WebSocket connection.
	5. The Charging Station sends a BootNotificationRequest	6. The OCTT responds with a BootNotificationResponse
	7. The Charging Station notifies the OCTT about the current state of all connectors.	8. The OCTT responds accordingly.
	<u>Note(s):</u> - Steps 5, 6, 7, and 8 are only required when status in Step 2 is <i>RebootRequired</i>	
Tool validations	* Step 2: Message: SetVariablesResponse - status must be Accepted or RebootRequired Post scenario validations: N/a	

Table 7. Test Case Id: TC_A_10_CS

Test case name	Update Charging Station Password for HTTP Basic Authentication - Rejected	
Test case Id	TC_A_10_CS	
Use case Id(s)	A01	
Requirement(s)	A01.FR.01, A01.FR.11, A01.FR.12	
System under test	Charging Station	
Description	This test case defines how to use the BasicAuthPassword, the password used to authenticate Charging Stations in security profile 1 (Basic Authentication) and security profile 2 (TLS with Basic Authentication)	
Purpose	To verify if the Charging Station is able to reject the new BasicAuthPassword.	
Prerequisite(s)	The charging station supports security profile 1 and/or 2	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with a SetVariablesResponse	<p>1. The OCTT sends a SetVariablesRequest</p> <p>setVariableData[1]:</p> <ul style="list-style-type: none"> - variable.name = "BasicAuthPassword" - component.name = "SecurityCtrlr" - attributeValue = "<NewPassword which is less than 16 characters>"
	3. The ChargingStation sends a HTTP upgrade request with an Authorization header, containing a username/password combination (with the old <i>BasicAuthPassword</i>).	<p>4. The OCTT validates the username/password combination AND upgrades the connection to a (secured) WebSocket connection.</p> <p>Note(s):</p> <ul style="list-style-type: none"> - The Authorization header is formatted as follows: <i>AUTHORIZATION: Basic <Base64 encoded(<Configured ChargingStationId>:<OLD BasicAuthPassword>)></i>
	5. Execute Reusable State <i>Booted</i>	
Tool validations	<ul style="list-style-type: none"> * Step 2: <p>Message: SetVariablesResponse</p> <ul style="list-style-type: none"> - status must be <i>Rejected</i> 	
	<p>Post scenario validations:</p> <p>BasicAuthPassword should be <Configured BasicAuthPassword> N/a</p>	

Table 8. Test Case Id: TC_A_11_CS

Test case name	Update Charging Station Certificate by request of CSMS - Success - Charging Station Certificate	
Test case Id	TC_A_11_CS	
Use case Id(s)	A02 & F06	
Requirement(s)	A02.FR.02, A02.FR.03, A02.FR.06, A02.FR.08, A02.FR.09 & F06.FR.04,F06.FR.05,F06.FR.10	
System under test	Charging Station	
Description	The CSMS is able to request the Charging Station to update its charging station certificate using the TriggerMessageRequest message.	
Purpose	To verify if the Charging Station is able to update its Charging Station Certificate.	
Prerequisite(s)	<ul style="list-style-type: none"> - The charging station supports security profile 3 - The active NetworkConnectionProfile uses security profile 3. 	
Before (Preparations)	<p>Configuration State: N/a</p> <p>Memory State: N/a</p> <p>Reusable State(s): N/a</p>	
Main (Test scenario)	Charging Station	CSMS
	1. Execute Reusable State RenewChargingStationCertificate for certificateType <i>ChargingStationCertificate</i>	
Tool validations	<p>N/a</p> <p>Post scenario validations: N/a</p>	

Table 9. Test Case Id: TC_A_14_CS

Test case name	Update Charging Station Certificate by request of CSMS - Invalid certificate	
Test case Id	TC_A_14_CS	
Use case Id(s)	A02	
Requirement(s)	A02.FR.07,A03.FR.07	
System under test	Charging Station	
Description	The CSMS is able to request the Charging Station to update its charging station certificate using the TriggerMessageRequest message.	
Purpose	To verify if the Charging Station is able to discard an invalid certificate and report a security event.	
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station supports security profile 3 - The active NetworkConnectionProfile uses security profile 3. 	
Before (Preparations)	<p>Configuration State: N/a</p> <p>Memory State: N/a</p> <p>Reusable State(s): N/a</p>	
Main (Test scenario)	<p>Charging Station</p> <p>2. The Charging Station responds with a TriggerMessageResponse</p> <p>3 The Charging Station sends a SignCertificateRequest</p> <p>6. The Charging Station responds with a CertificateSignedResponse</p> <p>7 The Charging Station sends a SecurityEventNotificationRequest</p>	<p>CSMS</p> <p>1. The OCTT sends a TriggerMessageRequest With requestedMessage <i>SignChargingStationCertificate</i></p> <p>4. The OCTT responds with a SignCertificateResponse With status Accepted</p> <p>5. The OCTT sends a CertificateSignedRequest With certificateChain <<i>Configured invalid_signingCertificate</i>> certificateType <i>ChargingStationCertificate</i></p> <p>8 The OCTT responds with a SecurityEventNotificationResponse</p>
Tool validations	<ul style="list-style-type: none"> * Step 2: Message: TriggerMessageResponse - status must be <i>Accepted</i> * Step 3: Message: SignCertificateRequest - csr must contain <An CSR that meets the following requirements: <i>When using RSA or DSA the key must be at least 2048 bits long.</i> <i>and when using elliptic curve cryptography the key must be at least 224 bits long.</i> <i>The received CSR must be transmitted as described in RFC 2986 and then encoded in Privacy-Enhanced Mail (PEM) format.></i> * Step 6: Message: CertificateSignedResponse - status must be <i>Rejected</i> * Step 7: Message: SecurityEventNotificationRequest - type must be <i>InvalidChargingStationCertificate</i> <p>Post scenario validations: N/a</p>	

Table 10. Test Case Id: TC_A_15_CS

Test case name	Update Charging Station Certificate by request of CSMS - SignCertificateRequest Rejected	
Test case Id	TC_A_15_CS	
Use case Id(s)	A02	
Requirement(s)	N/a	
System under test	Charging Station	
Description	The CSMS is able to request the Charging Station to update its charging station certificate using the TriggerMessageRequest message.	
Purpose	To verify if the Charging Station is able to discard an invalid certificate and report a security event.	
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station supports security profile 3 - The active NetworkConnectionProfile uses security profile 3. 	
Before (Preparations)	<p>Configuration State: N/a</p> <p>Memory State: N/a</p> <p>Reusable State(s): N/a</p>	
Main (Test scenario)	<p>Charging Station</p> <p>2. The Charging Station responds with a TriggerMessageResponse</p> <p>3 The Charging Station sends a SignCertificateRequest</p>	<p>CSMS</p> <p>1. The OCTT sends a TriggerMessageRequest With requestedMessage <i>SignChargingStationCertificate</i></p> <p>4. The OCTT responds with a SignCertificateResponse With status Rejected</p>
Tool validations	<p>* Step 2: Message: TriggerMessageResponse - status must be Accepted</p> <p>Post scenario validations: N/a</p>	

Table 11. Test Case Id: TC_A_23_CS

Test case name	Update Charging Station Certificate by request of CSMS - CertificateSignedRequest Timeout	
Test case Id	TC_A_23_CS	
Use case Id(s)	A02 & F06	
Requirement(s)	A02.FR.17,A02.FR.18	
System under test	Charging Station	
Description	The CSMS is able to request the Charging Station to update its charging station certificate using the TriggerMessageRequest message.	
Purpose	To verify if the Charging Station is able to send a new signCertificateRequest when it did not receive a certificateSignedRequest after the configured timeout.	
Prerequisite(s)	<ul style="list-style-type: none"> - The charging station supports security profile 3 - The Charging Station supports the CertificateSignedRequest Timeout feature 	
Before (Preparations)	<p>Configuration State: SecurityCtrlr.CertSigningWaitMinimum is <Configured CertSigningWaitMinimum> SecurityCtrlr.CertSigningRepeatTimes is 1</p> <p>Memory State: N/a</p> <p>Reusable State(s): N/a</p>	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with a TriggerMessageResponse	1. The OCTT sends a TriggerMessageRequest With requestedMessage <i>SignChargingStationCertificate</i>
	3 The Charging Station sends a SignCertificateRequest	4. The OCTT responds with a SignCertificateResponse With status Accepted
		5. The OCTT does NOT send the CertificateSignedRequest and waits for the SignCertificateRequest to be resend after the <Configured CertSigningWaitMinimum>
	6 The Charging Station sends a SignCertificateRequest	7. The OCTT responds with a SignCertificateResponse With status Accepted
		8. The OCTT does NOT send the CertificateSignedRequest and waits for the SignCertificateRequest to be resend after the <Configured CertSigningWaitMinimum> times 2
	9 The Charging Station sends a SignCertificateRequest	10. The OCTT responds with a SignCertificateResponse With status Accepted
	12. The Charging Station responds with a CertificateSignedResponse	11. The OCTT sends a CertificateSignedRequest With certificateChain <Certificate generated from the received CSR from step 3 and signed by the provided CSMS Root certificate> certificateType <i>ChargingStationCertificate</i>

Test case name	Update Charging Station Certificate by request of CSMS - CertificateSignedRequest Timeout
Tool validations	<ul style="list-style-type: none"> * Step 2: Message: TriggerMessageResponse - status must be Accepted * Step 3/6/9: Message: SignCertificateRequest - csr must contain <i><An CSR that meets the following requirements:</i> <i>When using RSA or DSA the key must be at least 2048 bits long.</i> <i>and when using elliptic curve cryptography the key must be at least 224 bits long.</i> <i>The received CSR must be transmitted as described in RFC 2986 and then encoded in Privacy-Enhanced Mail (PEM) format.></i> * Step 5: - The Charging Station shall not resend the SignCertificateRequest before the <i><Configured CertSigningWaitMinimum></i> expired * Step 8: - The Charging Station shall not resend the SignCertificateRequest before the <i><Configured CertSigningWaitMinimum></i> times 2 expired * Step 12: Message: CertificateSignedResponse - status must be Accepted <p>Post scenario validations: N/a</p>

Table 12. Test Case Id: TC_A_19_CS

Test case name	Upgrade Charging Station Security Profile - Accepted	
Test case Id	TC_A_19_CS	
Use case Id(s)	A05	
Requirement(s)	A05.FR.04,A05.FR.05,A05.FR.06	
System under test	Charging Station	
Description	The CSMS updates the connection details on the Charging Station, to increase the security profile level.	
Purpose	To verify if the Charging Station is able to increase the security profile level when configured to do so by the CSMS.	
Prerequisite(s)	Security profile must be set to 1 or 2	
Before (Preparations)	<p>Configuration State: N/a</p> <p>Memory State: If configured <Security profile> is 1, then <i>CertificateInstalled</i> If configured <Security profile> is 2, then <i>RenewChargingStationCertificate</i></p> <p>Reusable State(s): N/a</p>	
Main (Test scenario)	<p>Charging Station</p> <p>2. The Charging Station responds with a SetNetworkProfileResponse</p> <p>4. The Charging Station responds with a SetVariablesResponse</p> <p>6. The Charging Station responds with a ResetResponse</p> <p>7. The Charging Station reconnects to the OCTT with security profile is <Configured securityProfile + 1></p> <p>9. Execute Reusable State Booted</p> <p>11. The Charging Station responds with GetVariablesResponse</p> <p>13. The Charging Station responds with GetVariablesResponse</p>	<p>CSMS</p> <p>1. The OCTT sends a SetNetworkProfileRequest with configurationSlot is <Configured configurationSlot> or <Configured configurationSlot> depending on which one is already in use - connectionData.messageTimeout <Configured messageTimeout> - connectionData.ocppCsmsUrl <Configured ocppCsmsUrl> - connectionData.ocppInterface <Configured ocppInterface> - connectionData.ocppVersion OCPP20 - connectionData.securityProfile <Configured securityProfile + 1></p> <p>3. The OCTT sends a SetVariablesRequest with variable.name is "NetworkConfigurationPriority" component.name is "OCPPCommCtrlr" attributeValue is "<Configured configurationSlot + 1>,<Configured configurationSlot>"</p> <p>5. The OCTT sends a ResetRequest with type OnIdle</p> <p>Note(s): - This step will only be executed when the status RebootRequired is returned at step 4.</p> <p>8. The OCTT accepts the connection attempt.</p> <p>10. OCTT sends GetVariablesRequest with: - variable.name = "SecurityProfile" - component.name = "SecurityCtrlr"</p> <p>12. OCTT sends GetVariablesRequest with: - variable.name = "NetworkConfigurationPriority" - component.name = "OCPPCommCtrlr"</p>

Test case name	Upgrade Charging Station Security Profile - Accepted
Tool validations	<ul style="list-style-type: none"> * Step 2: Message SetNetworkProfileResponse - status Accepted * Step 4: Message SetVariablesResponse - setVariableResult[0].attributeStatus Accepted OR RebootRequired * Step 6: Message ResetResponse - status Accepted * Step 11: Message GetVariablesResponse - getVariableResult[0].attributeValue <Configured securityProfile + 1> * Step 13: Message GetVariablesResponse - getVariableResult[0].attributeValue Does not contain <Configured configurationSlot> <p>Post scenario validations:</p> <ul style="list-style-type: none"> - N/a

Table 13. Test Case Id: TC_A_20_CS

Test case name	Upgrade Charging Station Security Profile - No valid CSMSRootCertificate installed	
Test case Id	TC_A_20_CS	
Use case Id(s)	A05	
Requirement(s)	A05.FR.02	
System under test	Charging Station	
Description	The CSMS is able to change the connectionData at the Charging Station. By doing this it is able to upgrade the connection to a higher security profile.	
Purpose	To verify if the Charging Station is able to reject upgrading to a higher security profile when it does not have a valid CSMSRootCertificate installed.	
Prerequisite(s)	<ul style="list-style-type: none"> - The OCTT connectionData configuration for SUT Charging Station only allows for ip addresses the OCTT is able to bind. - The Charging Station support at least 2 security profiles, one of which is security profile 1. - The Charging Station does not have a valid CSMSRootCertificate installed. - The first OCTT connectionData configuration slot must be configured for security profile 1. - The second OCTT connectionData configuration slot must be configured for security profile 2 or 3. - When starting this testcase the OCTT will start another webSocket server for the second connectionData slot. 	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with a SetNetworkProfileResponse	<p>1. The OCTT sends a SetNetworkProfileRequest with - configurationSlot is <i><Configured configurationSlot></i> or <i><Configured configurationSlot></i> depending on which one is already in use</p> <ul style="list-style-type: none"> - connectionData.messageTimeout <i><Configured messageTimeout2></i> - connectionData.ocppCsmsUrl <i><ocppCsmsUrl that is not currently active></i> - connectionData.ocppInterface <i><Configured ocppInterface2></i> - connectionData.ocppVersion <i>OCPP20</i> - connectionData.securityProfile <i><Configured securityProfile2></i>
	4. The Charging Station responds with a SetVariablesResponse	<p>3. The OCTT sends a SetVariablesRequest with variable.name is "NetworkConfigurationPriority" component.name is "OCPPCommCtrlr" attributeValue is <i><Configured configurationSlot2></i>,<i><Configured configurationSlot></i></p>
Tool validations	<ul style="list-style-type: none"> * Step 2: Message SetNetworkProfileResponse - status Accepted * Step 4: Message SetVariablesResponse - setVariableResult[0].attributeStatus <i>Rejected</i> <p>Post scenario validations: - N/a</p>	

Table 14. Test Case Id: TC_A_21_CS

Test case name	Upgrade Charging Station Security Profile - No valid ChargingStationCertificate installed	
Test case Id	TC_A_21_CS	
Use case Id(s)	A05	
Requirement(s)	A05.FR.03	
System under test	Charging Station	
Description	The CSMS is able to change the connectionData at the Charging Station. By doing this it is able to upgrade the connection to a higher security profile.	
Purpose	To verify if the Charging Station is able to reject upgrading to a higher security profile when it does not have a valid ChargingStationCertificate installed.	
Prerequisite(s)	<ul style="list-style-type: none"> - The OCTT connectionData configuration for SUT Charging Station only allows for ip addresses the OCTT is able to bind. - The Charging Station support at least 2 security profiles. - The Charging Station does not have a valid ChargingStationCertificate installed. - The Charging Station has a valid CSMSRootCertificate installed. - The second OCTT connectionData configuration slot must be configured for security profile 3. - When starting this testcase the OCTT will start another webSocket server for the second connectionData slot. 	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with a SetNetworkProfileResponse	1. The OCTT sends a SetNetworkProfileRequest with - configurationSlot is <i><Configured configurationSlot></i> or <i><Configured configurationSlot></i> depending on which one is already in use - connectionData.messageTimeout <i><Configured messageTimeout2></i> - connectionData.ocppCsmsUrl <i><ocppCsmsUrl that is not currently active></i> - connectionData.ocppInterface <i><Configured ocppInterface2></i> - connectionData.ocppVersion <i>OCPP20</i> - connectionData.securityProfile <i><Configured securityProfile2></i>
	4. The Charging Station responds with a SetVariablesResponse	3. The OCTT sends a SetVariablesRequest with variable.name is "NetworkConfigurationPriority" component.name is "OCPPCommCtrlr" attributeValue is <i><Configured configurationSlot2>,<Configured configurationSlot></i>
Tool validations	* Step 2: Message SetNetworkProfileResponse - status Accepted * Step 4: Message SetVariablesResponse - setVariableResult[0].attributeStatus <i>Rejected</i> Post scenario validations: - N/a	

Table 15. Test Case Id: TC_A_22_CS

Test case name	Upgrade Charging Station Security Profile - Downgrade security profile - Rejected	
Test case Id	TC_A_22_CS	
Use case Id(s)	A05, B09	
Requirement(s)	B09.FR.04	
System under test	Charging Station	
Description	The CSMS is able to change the connectionData at the Charging Station. By doing this it is able to upgrade the connection to a higher security profile.	
Purpose	To verify if the Charging Station is able to reject upgrading to a higher security profile when it does not have a valid ChargingStationCertificate installed.	
Prerequisite(s)	<ul style="list-style-type: none"> - The OCTT connectionData configuration for SUT Charging Station only allows for ip addresses the OCTT is able to bind. - The Charging Station supports security profile 2 and/or 3. - The second OCTT connectionData configuration slot must be configured for a security profile lower than the first OCTT connectionData configuration slot. - When starting this testcase the OCTT will start another webSocket server for the second connectionData slot. 	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with a SetNetworkProfileResponse	<p>1. The OCTT sends a SetNetworkProfileRequest with:</p> <ul style="list-style-type: none"> - configurationSlot is <i><Configured configurationSlot></i> or <i><Configured configurationSlot></i> depending on which one is already in use - connectionData.messageTimeout <i><Configured messageTimeout2></i> - connectionData.ocppCsmsUrl <i><ocppCsmsUrl that is not currently active></i> - connectionData.ocppInterface <i><Configured ocppInterface2></i> - connectionData.ocppVersion <i>OCPP20</i> - connectionData.securityProfile <i><Configured securityProfile2></i>
Tool validations	<ul style="list-style-type: none"> * Step 2: Message SetNetworkProfileResponse - status <i>Rejected</i> 	
	Post scenario validations: <ul style="list-style-type: none"> - N/a 	

2.3. B Provisioning

Table 16. Test Case Id: TC_B_01_CS

Test case name	Cold Boot Charging Station - Accepted	
Test case Id	TC_B_01_CS	
Use case Id(s)	B01	
Requirement(s)	B01.FR.01, B01.FR.05, B01.FR.09	
System under test	Charging Station	
Description	The booting mechanism allows a Charging Station to provide some general information about the Charging Station to the CSMS on startup AND it allows the Charging Station to request whether it is allowed to start sending other OCPP messages.	
Purpose	To verify whether the Charging Station is able to perform the booting mechanism as described at the OCPP specification.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	1. Execute Reusable State <i>Booted</i>	
Tool validations	N/a	
	Post scenario validations: - A message to report the state of a connector has been received for all connectors.	

Table 17. Test Case Id: TC_B_02_CS

Test case name	Cold Boot Charging Station - Pending	
Test case Id	TC_B_02_CS	
Use case Id(s)	B02, F06	
Requirement(s)	B02.FR.01, B02.FR.02, B02.FR.04, B02.FR.05, B02.FR.06, B02.FR.08, F06.FR.17	
System under test	Charging Station	
Description	The booting mechanism allows a Charging Station to provide some general information about the Charging Station to the CSMS on startup AND it allows the Charging Station to request whether it is allowed to start sending other OCPP messages. A CSMS can temporarily halt the Charging Stations operations by returning the Pending status at the BootNotificationResponse. During this time the CSMS is able to retrieve and set configurations from the Charging Station.	
Purpose	To verify whether the Charging Station is able to correctly handle the pending state of the boot mechanism.	
Prerequisite(s)	The testcases; TC_B_06_CS, TC_B_09_CS, TC_B_13_CS are executed with test result PASS.	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	<u>Manual Action:</u> Reboot the Charging Station.	
	1. The Charging Station sends a BootNotificationRequest	2. The OCTT responds with a BootNotificationResponse with status Pending interval <Configured heartbeatInterval>
	4. The Charging Station responds with SetVariablesResponse	3. OCTT sends SetVariablesRequest with: - variable.name = "OfflineThreshold" - component.name = "OCPPCommCtrlr" - attributeValue = "300" - attributeType is omitted
	6. The Charging Station responds with GetVariablesResponse	5. OCTT sends GetVariablesRequest with: - variable.name = "OfflineThreshold" - component.name = "OCPPCommCtrlr" - attributeType is omitted
	8. Charging Station responds with: GetBaseReportResponse	7. OCTT sends GetBaseReportRequest with: - requestId = <Generated requestId> - reportBase = FullInventory

Test case name	Cold Boot Charging Station - Pending	
	Charging Station	CSMS
	9. Charging Station responds with: NotifyReportRequest	10. OCTT sends NotifyReportResponse
	<u>Note(s):</u> - This step is repeated as often as needed to report all configuration variables.	
	12. The Charging Station responds with a RequestStartTransactionResponse	11. The OCTT sends a RequestStartTransactionRequest
		<u>Note(s):</u> - This step is executed after the OCTT received all NotifyReport messages. This is indicated by the tbc and seqNo fields.
	14. The Charging Station responds with a TriggerMessageResponse	13. The OCTT sends a TriggerMessageRequest with requestedMessage BootNotification
	15. The Charging Station sends a BootNotificationRequest	16. The OCTT responds with a BootNotificationResponse with status Accepted interval <Configured heartbeatInterval>
	<u>Note(s):</u> - The Charging Station resends the BootNotificationRequest after having responded to the TriggerMessageRequest, so before the interval from the BootNotificationResponse has been passed.	
	17. The Charging Station notifies the CSMS about the current state of all connectors.	18. The OCTT responds accordingly.

Test case name	Cold Boot Charging Station - Pending
Tool validations	<ul style="list-style-type: none"> * Step 4: Message: SetVariablesResponse - setVariableResult[0].attributeStatus Accepted * Step 6: Message: GetVariablesResponse - getVariableResult[0].attributeStatus Accepted * Step 8: Message: GetBaseReportResponse - status Accepted * Step 12: Message: RequestStartTransactionResponse - status Rejected * Step 14: Message: TriggerMessageResponse - status Accepted or NotImplemented * Step 15: Message: BootNotificationRequest - reason Triggered (If the status from the response from step 14 contained Accepted) * Step 17: Message: StatusNotificationRequest - connectorStatus Available Message: NotifyEventRequest - eventData[0].trigger Delta - eventData[0].actualValue "Available" - eventData[0].component.name "Connector" - eventData[0].variable.name "AvailabilityState"
Post scenario validations:	- A message to report the state of a connector has been received for all connectors.

Table 18. Test Case Id: TC_B_03_CS

Test case name	Cold Boot Charging Station - Rejected	
Test case Id	TC_B_03_CS	
Use case Id(s)	B03	
Requirement(s)	B03.FR.02, B03.FR.04, B03.FR.06	
System under test	Charging Station	
Description	The booting mechanism allows a Charging Station to provide some general information about the Charging Station to the CSMS on startup AND it allows the Charging Station to request whether it is allowed to start sending other OCPP messages.	
Purpose	To verify whether the Charging Station is able to correctly handle a rejected BootNotification.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): N/a	
Main (Test scenario)	Charging Station <u>Manual Action:</u> Reboot the Charging Station.	CSMS
	1. The Charging Station sends a BootNotificationRequest	2. The OCTT responds with a BootNotificationResponse with status Rejected interval <Configured heartbeatInterval>
	3. The Charging Station sends a BootNotificationRequest	4. The OCTT responds with a BootNotificationResponse with status Accepted interval <Configured heartbeatInterval>
	<u>Note(s):</u> - The Charging Station resends the BootNotificationRequest after x seconds, whereby x is equal to or greater than the interval from the BootNotificationResponse. - The Charging Station is not allowed to send any OCPP message in the meantime. - The Charging Station is allowed to close the connection until it needs to resend the BootNotificationRequest.	
	5. The Charging Station notifies the CSMS about the current state of all connectors.	6. The OCTT responds accordingly.
Tool validations	* Step 5: Message: StatusNotificationRequest - connectorStatus Available Message: NotifyEventRequest - eventData[0].trigger Delta - eventData[0].actualValue "Available" - eventData[0].component.name "Connector" - eventData[0].variable.name "AvailabilityState" Post scenario validations: - A message to report the state of a connector has been received for all connectors.	

Table 19. Test Case Id: TC_B_30_CS

Test case name	Cold Boot Charging Station - Pending/Rejected - SecurityError	
Test case Id	TC_B_30_CS	
Use case Id(s)	B03	
Requirement(s)	B03.FR.08	
System under test	Charging Station	
Description	The booting mechanism allows a Charging Station to provide some general information about the Charging Station to the CSMS on startup AND it allows the Charging Station to request whether it is allowed to start sending other OCPP messages. The CSMS may respond to the BootNotificationRequest with status <i>Rejected</i> . During this state, the Charging Station is not allowed to send RPC Framework: CALL message that is NOT a BootNotificationRequest.	
Purpose	To verify whether the Charging Station is able to handle unauthorized messages from the CSMS by responding with a SecurityError.	
Prerequisite(s)	The Charging Station is configured to keep the connection open while it is waiting to resend the BootNotificationRequest.	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	1. The Charging Station sends a BootNotificationRequest	2. The OCTT responds with a BootNotificationResponse with status <i>Rejected</i>
	4. The Charging Station responds with RPC Framework: CALLERROR: SecurityError.	3. The OCTT sends a GetBaseReportRequest with reportBase FullInventory
Tool validations	N/a	
	N/a	

Table 20. Test Case Id: TC_B_06_CS

Test case name	Get Variables - single value	
Test case Id	TC_B_06_CS	
Use case Id(s)	B06	
Requirement(s)	B06.FR.01, B06.FR.02, B06.FR.03, B06.FR.04, B06.FR.10, B06.FR.11	
System under test	Charging Station	
Description	Get the value of one of the required variables of OCPPCommCtrlr	
Purpose	To test getting a single value using GetVariablesRequest for one of the mandatory component/variable combinations that must exist in the DM implementation.	
Prerequisite(s)	N/A	
Before (Preparations)	<p>Configuration State: OCPPCommCtrlr.OfflineThreshold is 300</p> <p>Memory State: N/a</p> <p>Reusable State(s): N/a</p>	
Main (Test scenario)	<p>Charging Station</p> <p>2. Charging Station responds with GetVariablesResponse</p>	<p>CSMS</p> <p>1. OCTT sends GetVariablesRequest with: - variable.name = "OfflineThreshold" - component.name = "OCPPCommCtrlr" - attributeType = Actual</p>
Tool validations	<p>* Step 2: Message: GetVariablesResponse - attributeStatus = Accepted - attributeType = Actual - attributeValue = "300" - component.name = "OCPPCommCtrlr" - variable.name = "OfflineThreshold" - attributeStatusInfo is absent or attributeStatusInfo.reasonCode = "NoError"</p> <p>Post scenario validations: N/A</p>	

Table 21. Test Case Id: TC_B_07_CS

Test case name	Get Variables - multiple values	
Test case Id	TC_B_07_CS	
Use case Id(s)	B06	
Requirement(s)	B06.FR.01, B06.FR.02, B06.FR.03, B06.FR.04, B06.FR.10	
System under test	Charging Station	
Description	Get the value of two required variables	
Purpose	To test getting multiple values using GetVariablesRequest for one of the mandatory component/variable combinations that must exist in the DM implementation.	
Prerequisite(s)	N/A	
Before (Preparations)	<p>Configuration State: OCPPCommCtrlr.OfflineThreshold is 300 AuthCtrlr.LocalAuthorizeOffline is true</p> <p>Memory State: N/a</p> <p>Reusable State(s): N/a</p>	
Main (Test scenario)	<p>Charging Station</p> <p>2. Charging Station responds with GetVariablesResponse with attributeStatus = Accepted.</p>	<p>CSMS</p> <p>1. OCTT sends GetVariablesRequest with:</p> <ul style="list-style-type: none"> - getVariableData[0].variable.name = "OfflineThreshold" - getVariableData[0].component.name = "OCPPCommCtrlr" - getVariableData[0].attributeType = Actual - getVariableData[1].variable.name = "LocalAuthorizeOffline" - getVariableData[1].component.name = "AuthCtrlr" - getVariableData[1].attributeType = Actual
Tool validations	<p>* Step 2:</p> <p>Message: GetVariablesResponse has (in arbitrary order)</p> <p>GetVariableResultType[0]:</p> <ul style="list-style-type: none"> - attributeStatus = Accepted - attributeType = Actual - attributeValue = 300 - component.name = "OCPPCommCtrlr" - variable.name = "OfflineThreshold" - attributeStatusInfo is absent or attributeStatusInfo.reasonCode = "NoError" <p>GetVariableResultType[1]:</p> <ul style="list-style-type: none"> - attributeStatus = Accepted - attributeType = Actual - attributeValue = "true" - component.name = "AuthCtrlr" - variable.name = "LocalAuthorizeOffline" - attributeStatusInfo is absent or attributeStatusInfo.reasonCode = "NoError" <p>Post scenario validations: N/A</p>	

Table 22. Test Case Id: TC_B_32_CS

Test case name	Get Variables - Unknown component	
Test case Id	TC_B_32_CS	
Use case Id(s)	B06	
Requirement(s)	B06.FR.06	
System under test	Charging Station	
Description	The CSMS can use a GetVariablesRequest to retrieve values from device model variables at the Charging Station.	
Purpose	To verify whether the Charging Station can handle receiving a GetVariablesRequest for an unknown component.	
Prerequisite(s)	N/A	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with GetVariablesResponse	1. OCTT sends GetVariablesRequest with: - variable.name = "OfflineThreshold" - component.name = "UnknownComponent" - attributeType is omitted
Tool validations	* Step 2: Message: GetVariablesResponse - getVariableResult[0].attributeStatus = UnknownComponent - getVariableResult[0].component.name = "UnknownComponent" - getVariableResult[0].variable.name = "OfflineThreshold" Post scenario validations: N/A	

Table 23. Test Case Id: TC_B_33_CS

Test case name	Get Variables - Unknown variable	
Test case Id	TC_B_33_CS	
Use case Id(s)	B06	
Requirement(s)	B06.FR.07	
System under test	Charging Station	
Description	The CSMS can use a GetVariablesRequest to retrieve values from device model variables at the Charging Station.	
Purpose	To verify whether the Charging Station can handle receiving a GetVariablesRequest for an unknown variable.	
Prerequisite(s)	N/A	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with GetVariablesResponse	1. OCTT sends GetVariablesRequest with: - variable.name = "UnknownVariable" - component.name = "OCPPCommCtrlr" - attributeType is omitted
Tool validations	* Step 2: Message: GetVariablesResponse - getVariableResult[0].attributeStatus = UnknownVariable - getVariableResult[0].component.name = "OCPPCommCtrlr" - getVariableResult[0].variable.name = "UnknownVariable"	
	Post scenario validations: N/A	

Table 24. Test Case Id: TC_B_34_CS

Test case name	Get Variables - Not supported attribute type	
Test case Id	TC_B_34_CS	
Use case Id(s)	B06	
Requirement(s)	B06.FR.08	
System under test	Charging Station	
Description	The CSMS can use a GetVariablesRequest to retrieve values from device model variables at the Charging Station.	
Purpose	To verify whether the Charging Station can handle receiving a GetVariablesRequest for a not supported attribute type.	
Prerequisite(s)	N/A	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with GetVariablesResponse	1. OCTT sends GetVariablesRequest with: - variable.name = "OfflineThreshold" - component.name = "OCPPCommCtrlr" - attributeType = Target
Tool validations	* Step 2: Message: GetVariablesResponse - getVariableResult[0].attributeStatus = NotSupportedAttributeType - getVariableResult[0].component.name = "OCPPCommCtrlr" - getVariableResult[0].variable.name = "OfflineThreshold" - getVariableResult[0].attributeType = Target Post scenario validations: N/A	

Table 25. Test Case Id: TC_B_08_CS

Test case name	Get Variables - limit to maximum number of values	
Test case Id	TC_B_08_CS	
Use case Id(s)	B06	
Requirement(s)	B06.FR.05	
System under test	Charging Station	
Description	Do not return more variables than supported by <code>MaxItemsPerMessageGetVariables</code> .	
Purpose	To test that Charging Station does not return more variables than it reports to support in the variable <code>MaxItemsPerMessageGetVariables</code> .	
Prerequisite(s)	CS needs to have more configured variables than <code>MaxItemsPerMessageGetVariables</code> .	
Before (Preparations)	<p>Configuration State: N/a</p> <p>Memory State: N/a</p> <p>Reusable State(s): N/a</p>	
Main (Test scenario)	<p>Charging Station</p> <p>2. Charging Station responds with GetBaseReportResponse</p> <p>3. Charging Station sends NotifyReportRequest</p> <p>Note(s): - This step may be executed multiple times, until all components/variables are reported.</p> <p>6. Charging Station responds with GetVariablesResponse</p> <p>Note(s): - It is up to Charging Station to decide whether it wants to return every variable as Rejected or return <code>ItemsPerMessageGetVariables</code> values with attributeStatus = Accepted and the rest as Rejected.</p>	<p>CSMS</p> <p>1. OCTT sends GetBaseReportRequest for type <code>FullInventory</code></p> <p>4. OCTT responds with NotifyReportResponse</p> <p>5. OCTT sends GetVariablesRequest for x amount of variables:</p> <p>Note(s): - x equals <code>ItemsPerMessageGetVariables + 1</code></p>
Tool validations	<ul style="list-style-type: none"> * Step 6: Message: GetVariablesResponse has a list of GetVariableResultType values (in arbitrary order) of which either: <ul style="list-style-type: none"> - all have attributeStatus = Rejected - or the last has attributeStatus = Rejected and the rest have attributeStatus = Accepted. and if attributeStatusInfo is provided: <ul style="list-style-type: none"> - the accepted items have attributeStatusInfo.reasonCode = "NoError" - the rejected items have attributeStatusInfo.reasonCode = "TooManyElements" <p>Post scenario validations: N/A</p>	

Table 26. Test Case Id: TC_B_09_CS

Test case name	Set Variables - single value	
Test case Id	TC_B_09_CS	
Use case Id(s)	B05	
Requirement(s)	B05.FR.01, B05.FR.02, B05.FR.03, B05.FR.10, B05.FR.12	
System under test	Charging Station	
Description	Set the value of one of the required variables of OCPPCommCtrlr	
Purpose	To test setting a single value using SetVariablesRequest for one of the mandatory component/variable combinations that must exist in the DM implementation.	
Prerequisite(s)	N/A	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): N/a	
Main (Test scenario)	Charging Station 2. Charging Station responds with SetVariablesResponse with attributeStatus = Accepted.	CSMS 1. OCTT sends SetVariablesRequest with: <ul style="list-style-type: none"> - variable.name = "OfflineThreshold" - component.name = "OCPPCommCtrlr" - attributeValue = "300" - attributeType Actual
Tool validations	* Step 2: Message: SetVariablesResponse <ul style="list-style-type: none"> - setVariableResult[0].attributeStatus = Accepted - setVariableResult[0].attributeType = Actual - setVariableResult[0].component.name = "OCPPCommCtrlr" - setVariableResult[0].variable.name = "OfflineThreshold" - setVariableResult[0].attributeStatusInfo is absent or setVariableResult[0].attributeStatusInfo.reasonCode = "NoError" Post scenario validations: N/A	

Table 27. Test Case Id: TC_B_10_CS

Test case name	Set Variables - multiple values	
Test case Id	TC_B_10_CS	
Use case Id(s)	B05	
Requirement(s)	B05.FR.01, B05.FR.02, B05.FR.03, B05.FR.10, B05.FR.12	
System under test	Charging Station	
Description	Set the value of two required variables	
Purpose	To test setting multiple values using GetVariablesRequest for one of the mandatory component/variable combinations that must exist in the DM implementation.	
Prerequisite(s)	N/A	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	2. Charging Station responds with SetVariablesResponse with attributeStatus = Accepted.	<p>1. OCTT sends SetVariablesRequest with:</p> <ul style="list-style-type: none"> - setVariableData[0].variable.name = "OfflineThreshold" - setVariableData[0].component.name = "OCPPCommCtrlr" - setVariableData[0].attributeValue = "300" - setVariableData[0].attributeType = Actual - setVariableData[1].variable.name = "LocalAuthorizeOffline" - setVariableData[1].component.name = "AuthCtrlr" - setVariableData[1].attributeValue = "true" - setVariableData[0].attributeType = Actual
Tool validations	<p>* Step 2:</p> <p>Message: SetVariablesResponse has (in arbitrary order)</p> <p>SetVariableResultType[1]:</p> <ul style="list-style-type: none"> - attributeStatus = Accepted - attributeType = Actual - component.name = "OCPPCommCtrlr" - variable.name = "OfflineThreshold" - attributeStatusInfo is absent or attributeStatusInfo.reasonCode = "NoError" <p>SetVariableResultType[2]:</p> <ul style="list-style-type: none"> - attributeStatus = Accepted - attributeType = Actual - component.name = "AuthCtrlr" - variable.name = "LocalAuthorizeOffline" - attributeStatusInfo is absent or attributeStatusInfo.reasonCode = "NoError" <p>Post scenario validations:</p> <p>N/A</p>	

Table 28. Test Case Id: TC_B_35_CS

Test case name	Set Variables - Unknown component	
Test case Id	TC_B_35_CS	
Use case Id(s)	B05	
Requirement(s)	B05.FR.04	
System under test	Charging Station	
Description	The CSMS can use a SetVariablesRequest to set values from device model variables at the Charging Station.	
Purpose	To verify whether the Charging Station can handle receiving a SetVariablesRequest for an unknown component.	
Prerequisite(s)	N/A	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with SetVariablesResponse	1. OCTT sends SetVariablesRequest with: - variable.name = "OfflineThreshold" - component.name = "UnknownComponent" - attributeType is omitted
Tool validations	<ul style="list-style-type: none"> * Step 2: Message: SetVariablesResponse - setVariableResult[0].attributeStatus = UnknownComponent - setVariableResult[0].component.name = "UnknownComponent" - setVariableResult[0].variable.name = "OfflineThreshold" 	
	Post scenario validations: N/A	

Table 29. Test Case Id: TC_B_36_CS

Test case name	Set Variables - Unknown variable	
Test case Id	TC_B_36_CS	
Use case Id(s)	B05	
Requirement(s)	B05.FR.05	
System under test	Charging Station	
Description	The CSMS can use a SetVariablesRequest to set values from device model variables at the Charging Station.	
Purpose	To verify whether the Charging Station can handle receiving a SetVariablesRequest for an unknown variable.	
Prerequisite(s)	N/A	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with SetVariablesResponse	1. OCTT sends SetVariablesRequest with: <ul style="list-style-type: none"> - variable.name = "UnknownVariable" - component.name = "OCPPCommCtrlr" - attributeType is omitted
Tool validations	* Step 2: Message: SetVariablesResponse <ul style="list-style-type: none"> - setVariableResult[0].attributeStatus = UnknownVariable - setVariableResult[0].component.name = "OCPPCommCtrlr" - setVariableResult[0].variable.name = "UnknownVariable" 	
	Post scenario validations: N/A	

Table 30. Test Case Id: TC_B_37_CS

Test case name	Set Variables - Not supported attribute type	
Test case Id	TC_B_37_CS	
Use case Id(s)	B05	
Requirement(s)	B05.FR.06	
System under test	Charging Station	
Description	The CSMS can use a SetVariablesRequest to set values from device model variables at the Charging Station.	
Purpose	To verify whether the Charging Station can handle receiving a SetVariablesRequest for a not supported attribute type.	
Prerequisite(s)	N/A	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with SetVariablesResponse	1. OCTT sends SetVariablesRequest with: - variable.name = "OfflineThreshold" - component.name = "OCPPCommCtrlr" - attributeType = Target
Tool validations	* Step 2: Message: SetVariablesResponse - setVariableResult[0].attributeStatus = NotSupportedAttributeType - setVariableResult[0].component.name = "OCPPCommCtrlr" - setVariableResult[0].variable.name = "OfflineThreshold" - setVariableResult[0].attributeType = Target Post scenario validations: N/A	

Table 31. Test Case Id: TC_B_11_CS

Test case name	Set Variables - invalidly formatted values
Test case Id	TC_B_11_CS
Use case Id(s)	B05
Requirement(s)	B05.FR.07
System under test	Charging Station
Description	Set the value of two of the required variables of OCPPCommCtrlr
Purpose	To test setting of variables of different type with invalidly formatted values using GetVariablesRequest for one of the mandatory component/variable combinations that must exist in the DM implementation.
Prerequisite(s)	Charging Station DM has the variable "NextTimeOffsetTransitionDate" of component "ClockCtrlr" to test setting of a date.
Before (Preparations)	<p>Configuration State: N/a</p> <p>Memory State: N/a</p> <p>Reusable State(s): N/a</p>

Test case name	Set Variables - invalidly formatted values	
Main (Test scenario)	Charging Station	CSMS
<u>Notes:</u> Steps 1 to 8 are repeated 5 times for value = 1, 1.1, true, currentTime, "abc"		
	2. Charging Station responds with SetVariablesResponse with <i>If component/variable/value not supported:</i> attributeStatus = Rejected attributeStatusInfo = InvalidValue <i>If component/variable/value supported:</i> attributeStatus = Accepted	1. OCTT sends SetVariablesRequest with <ul style="list-style-type: none"> - variable.name = "OfflineThreshold" - component.name = "OCPPCommCtrlr" - attributeValue = value
	4. Charging Station responds with SetVariablesResponse with <i>If component/variable/value not supported:</i> attributeStatus = Rejected attributeStatusInfo = InvalidValue <i>If component/variable/value supported:</i> attributeStatus = Accepted	3. OCTT sends SetVariablesRequest with <ul style="list-style-type: none"> - variable.name = "LimitChangeSignificance" - component.name = "SmartChargingCtrlr" - attributeValue = value
<u>Notes:</u> Steps 5 and 6 will only be executed if this component/variable combination is readwrite		
	6. Charging Station responds with SetVariablesResponse with <i>If component/variable/value not supported:</i> attributeStatus = Rejected attributeStatusInfo = InvalidValue <i>If component/variable/value supported:</i> attributeStatus = Accepted	5. OCTT sends SetVariablesRequest with: <ul style="list-style-type: none"> - variable.name = "AuthorizeRemoteStart" - component.name = "AuthCtrlr" - attributeValue = value
<u>Notes:</u> Steps 7 and 8 will only be executed if the CS supports this component/variable combination		
	8. Charging Station responds with SetVariablesResponse with <i>If component/variable/value not supported:</i> attributeStatus = Rejected attributeStatusInfo = InvalidValue <i>If component/variable/value supported:</i> attributeStatus = Accepted	7. OCTT sends SetVariablesRequest with: <ul style="list-style-type: none"> - variable.name = "NextTimeOffsetTransitionDateTime" - component.name = "ClockCtrlr" - attributeValue = value

Test case name	Set Variables - invalidly formatted values
Tool validations	<ul style="list-style-type: none"> * Step 2: Message: SetVariablesResponse has SetVariableResultType - attributeStatus = Rejected/Accepted - attributeType = Actual - component.name = "OCPPCommCtrlr" - variable.name = "OfflineThreshold" - attributeStatusInfo is absent or attributeStatusInfo.reasonCode = <i>InvalidValue</i> (not required)
	<ul style="list-style-type: none"> * Step 4: Message: SetVariablesResponse has SetVariableResultType - attributeStatus = Rejected/Accepted - attributeType = Actual - component.name = "AuthCtrlr" - variable.name = "AuthorizeRemoteStart" - attributeStatusInfo is absent or attributeStatusInfo.reasonCode = <i>InvalidValue</i> (not required)
	<ul style="list-style-type: none"> * Step 6: Message: SetVariablesResponse has SetVariableResultType - attributeStatus = Rejected/Accepted - attributeType = Actual - component.name = "SmartChargingCtrlr" - variable.name = "LimitChangeSignificance" - attributeStatusInfo is absent or attributeStatusInfo.reasonCode = <i>InvalidValue</i> (not required)
	<ul style="list-style-type: none"> * Step 8: Message: SetVariablesResponse has SetVariableResultType - attributeStatus = Rejected/Accepted - attributeType = Actual - component.name = "ClockCtrlr" - variable.name = "NextTimeOffsetTransitionDateTime" - attributeStatusInfo is absent or attributeStatusInfo.reasonCode = <i>InvalidValue</i> (not required)
	<p>Post scenario validations: N/A</p>

Table 32. Test Case Id: TC_B_39_CS

Test case name	Set Variables - Read-only	
Test case Id	TC_B_39_CS	
Use case Id(s)	B05	
Requirement(s)	B05.FR.09	
System under test	Charging Station	
Description	The CSMS can use a SetVariablesRequest to set values from device model variables at the Charging Station.	
Purpose	To verify whether the Charging Station can handle receiving a SetVariablesRequest for a Read-only variable.	
Prerequisite(s)	N/A	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with SetVariablesResponse	1. OCTT sends SetVariablesRequest with: - variable.name = "MessageTimeout" - variable.instance = "Default" - component.name = "OCPPCommCtrlr" - attributeType is omitted
Tool validations	* Step 2: Message: SetVariablesResponse - setVariableResult[0].attributeStatus = Rejected - setVariableResult[0].component.name = "OCPPCommCtrlr" - setVariableResult[0].variable.name = "MessageTimeout" - setVariableResult[0].variable.instance = "Default"	
	Post scenario validations: N/A	

Table 33. Test Case Id: TC_B_12_CS

Test case name	Get Base Report - ConfigurationInventory	
Test case Id	TC_B_12_CS	
Use case Id(s)	B07	
Requirement(s)	B07.FR.01, B07.FR.03, B07.FR.04, B07.FR.07 , B07.FR.10, B07.FR.12	
System under test	Charging Station	
Description	CSMS requests a ConfigurationInventory base report.	
Purpose	To test that Charging Station supports the ConfigurationInventory base report.	
Prerequisite(s)	N/A	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	2. Charging Station responds with: GetBaseReportResponse	1. OCTT sends GetBaseReportRequest with: - requestId = <Generated requestId> - reportBase = ConfigurationInventory
	3. Charging Station responds with: NotifyReportRequest	4. OCTT sends NotifyReportResponse
	Step 3 and 4 are repeated as often as needed to report all configuration variables.	
Tool validations	<ul style="list-style-type: none"> * Step 2: <p>Message: GetBaseReportResponse</p> <ul style="list-style-type: none"> - status = Accepted - statusInfo is absent or statusInfo.reasonCode = "NoError" 	
	<ul style="list-style-type: none"> * Step 3: <p>Message: NotifyReportRequest</p> <ul style="list-style-type: none"> - requestId = <Generated requestId> - generatedAt = <timestamp at charging station> - seqNo = 0 - if variableCharacteristics.dataType = OptionList, SequenceList or MemberList then valuesList must be provided. 	
	while tbc = true	Expect NotifyReportRequest - seqNo is incremented by 1
	Post scenario validations: Check for all received variables: - variableCharacteristics are present - mutability = ReadWrite or WriteOnly Validate that as a minimum the required writable variables in section "Referenced Components and Variables" are reported, that are relevant to each functional block that has been implemented.	

Table 34. Test Case Id: TC_B_13_CS

Test case name	Get Base Report - FullInventory	
Test case Id	TC_B_13_CS	
Use case Id(s)	B07	
Requirement(s)	B07.FR.01, B07.FR.03, B07.FR.04, B07.FR.08 , B07.FR.10, B07.FR.12	
System under test	Charging Station	
Description	CSMS requests a FullInventory base report.	
Purpose	To test that Charging Station supports the FullInventory base report.	
Prerequisite(s)	N/A	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	2. Charging Station responds with: GetBaseReportResponse	1. OCTT sends GetBaseReportRequest with: - requestId = <Generated requestId> - reportBase = <i>FullInventory</i>
	3. Charging Station responds with: NotifyReportRequest	4. OCTT sends NotifyReportResponse
	Step 3 and 4 are repeated as often as needed to report all configuration variables.	
Tool validations	<ul style="list-style-type: none"> * Step 2: <p>Message: GetBaseReportResponse</p> <ul style="list-style-type: none"> - status = Accepted - statusInfo is absent or statusInfo.reasonCode = "NoError" 	
	<ul style="list-style-type: none"> * Step 3: <p>Message: NotifyReportRequest</p> <ul style="list-style-type: none"> - requestId = <Generated requestId> - generatedAt = <timestamp at charging station> - seqNo = 0 - if variableCharacteristics.dataType = <i>OptionList</i>, <i>SequenceList</i> or <i>MemberList</i> then valuesList must be provided. 	
	while tbc = true	Expect NotifyReportRequest - seqNo is incremented by 1
	Post scenario validations:	
	<p>Check for all received variables:</p> <ul style="list-style-type: none"> - variableCharacteristics are present <p>Validate that as a minimum the required variables mentioned in section "Charging Infrastructure Related" are reported as well as the required variables in section "Referenced Components and Variables", that are relevant to each functional block that has been implemented.</p>	

Table 35. Test Case Id: TC_B_15_CS

Test case name	Get Base Report - Not Supported base report	
Test case Id	TC_B_15_CS	
Use case Id(s)	B07	
Requirement(s)	B07.FR.02	
System under test	Charging Station	
Description	CSMS requests a base report that is not supported.	
Purpose	To test that Charging Station returns NotSupported when a SummaryInventory base report is requested, but Charging Station does not support it.	
Prerequisite(s)	Charging Station implementation does not support the optional SummaryInventory report.	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	2. Charging Station responds with: GetBaseReportResponse	1. OCTT sends GetBaseReportRequest with: - requestId = <Generated requestId> - reportBase = <i>SummaryInventory</i>
	Note(s): - OCTT waits to make sure CS does not send a NotifyReportRequest	
Tool validations	* Step 2: Charging Station responds with: GetBaseReportResponse with: - status = <i>NotSupported</i> - statusInfo is absent or statusInfo.reasonCode = " <i>UnsupportedParam</i> "	
	Post scenario validations: N/A	

Table 36. Test Case Id: TC_B_20_CS

Test case name	Reset Charging Station - Without ongoing transaction - OnIdle	
Test case Id	TC_B_20_CS	
Use case Id(s)	B11	
Requirement(s)	B11.FR.01, B11.FR.03, B11.FR.04, B01.FR.03	
System under test	Charging Station	
Description	This test case covers how the CSMS can request the Charging Station to reset itself by sending a ResetRequest without any ongoing transaction. This could for example be necessary if the Charging Station is not functioning correctly.	
Purpose	To verify if the Charging Station is able to perform the reset mechanism as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): N/a	
Main (Test scenario)	Charging Station 2. The Charging Station responds with a ResetResponse Note(s): - Charging Station reboots	CSMS 1. The OCTT sends a ResetRequest with type OnIdle
	 3. The Charging Station sends a BootNotificationRequest	 4. The OCTT responds with a BootNotificationResponse with status Accepted
	 5. The Charging Station notifies the CSMS about the current state of all connectors.	 6. The OCTT responds accordingly.
Tool validations	<ul style="list-style-type: none"> * Step 2: Message ResetResponse - status Accepted * Step 5: Message: StatusNotificationRequest - connectorStatus Available Message: NotifyEventRequest - eventData[0].trigger Delta - eventData[0].actualValue "Available" - eventData[0].component.name "Connector" - eventData[0].variable.name "AvailabilityState" <p>Post scenario validations: - A message to report the state of a connector has been received for all connectors.</p>	

Table 37. Test Case Id: TC_B_21_CS

Test case name	Reset Charging Station - With Ongoing Transaction - OnIdle	
Test case Id	TC_B_21_CS	
Use case Id(s)	B12	
Requirement(s)	B12.FR.01, B12.FR.03, E07.FR.03	
System under test	Charging Station	
Description	<p>This test case covers how the CSMS can remotely request the Charging Station to reset itself by sending a ResetRequest during a transaction. When ResetRequest "OnIdle" is send the charging stations schedules a reboot after all transactions are stopped.</p> <p>This could for example be necessary if the Charging Station is not functioning correctly.</p>	
Purpose	To verify if the Charging Station is able to perform the reset mechanism while there is an ongoing transaction as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	<p>Configuration State: N/a</p> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>EnergyTransferStarted</i></p>	
Main (Test scenario)	<p>Charging Station</p> <p>2. The Charging Station responds with a ResetResponse</p> <p>3. Execute Reusable State <i>StopAuthorized</i></p> <p>4. Execute Reusable State <i>EVConnectedPostSession</i></p> <p>5. Execute Reusable State <i>EVDDisconnected</i></p> <p><u>Notes(s): Steps 4 and 5 will only be executed if TxStartPoint does not contain: EnergyTransferStarted, DataSigned, PowerPathClosed, or Authorized</u></p> <p>6. Execute Reusable State <i>ParkingBayUnoccupied</i></p> <p><u>Notes(s): Step 6 will only be executed if TxStartPoint does not contain: EnergyTransferStarted, DataSigned, PowerPathClosed, Authorized, or EVConnected</u></p> <p>7. The Charging Station sends a BootNotificationRequest</p> <p>9. The Charging Station notifies the CSMS about the current state of all connectors.</p> <p>11. The Charging Station sends a SecurityEventNotificationRequest</p>	<p>CSMS</p> <p>1. The OCTT sends a ResetRequest with type OnIdle</p> <p>8. The OCTT responds with a BootNotificationResponse</p> <p>10. The OCTT responds accordingly.</p> <p>12. The OCTT responds with a SecurityEventNotificationResponse</p>

Test case name	Reset Charging Station - With Ongoing Transaction - OnIdle
Tool validations	<ul style="list-style-type: none"> * Step 2: Message ResetResponse - status <i>Scheduled</i> * Step 7: Message BootNotificationRequest - reason <i>ScheduledReset</i> * Step 9: Message: StatusNotificationRequest - If the transaction was stopped at step 3, then connectorStatus <i>Occupied</i> Else connectorStatus <i>Available</i> Message: NotifyEventRequest - If the transaction was stopped at step 3, then eventData[0].actualValue "Occupied" Else eventData[0].actualValue "Available" - eventData[0].trigger <i>Delta</i> - eventData[0].component.name "Connector" - eventData[0].variable.name "AvailabilityState" * Step 11: Message: SecurityEventNotificationRequest - type <i>StartupOfTheDevice or ResetOrReboot</i>
	<p>Post scenario validations:</p> <ul style="list-style-type: none"> - A message to report the state of a connector has been received for all connectors.

Table 38. Test Case Id: TC_B_22_CS

Test case name	Reset Charging Station - With Ongoing Transaction - Immediate	
Test case Id	TC_B_22_CS	
Use case Id(s)	B12	
Requirement(s)	B12.FR.02, B12.FR.04, E07.FR.03, B01.FR.03	
System under test	Charging Station	
Description	<p>This test case covers how the CSMS can remotely request the Charging Station to reset itself by sending a ResetRequest during a transaction. When ResetRequest "Immediate" is send the charging stations will try to stop all transactions before rebooting.</p> <p>This could for example be necessary if the Charging Station is not functioning correctly.</p>	
Purpose	To verify if the Charging Station is able to perform the reset mechanism as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): State is <i>EnergyTransferStarted</i>	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with a ResetResponse	1. The OCTT sends a ResetRequest with type Immediate
	3. The Charging Station sends a TransactionEventRequest	4. The OCTT responds with a TransactionEventResponse
	<u>Note(s):</u> - Charging Station reboots	
	5. The Charging Station sends a BootNotificationRequest	6. The OCTT responds with a BootNotificationResponse with status Accepted
	7. The Charging Station notifies the CSMS about the current state of all connectors.	8. The OCTT responds accordingly.

Test case name	Reset Charging Station - With Ongoing Transaction - Immediate
Tool validations	<ul style="list-style-type: none"> * Step 2: Message ResetResponse - status Accepted * Step 3: Message TransactionEventRequest - eventType Ended - triggerReason ResetCommand - transactionInfo.chargingState EVConnected - transactionInfo.stoppedReason ImmediateReset - idToken must be omitted * Step 5: Message BootNotificationRequest - reason RemoteReset * Step 7: For <Configured connectorId>: Message: StatusNotificationRequest - connectorStatus Occupied Message: NotifyEventRequest - eventData[0].trigger Delta - eventData[0].actualValue "Occupied" - eventData[0].component.name "Connector" - eventData[0].variable.name "AvailabilityState" For <Other connector(s)>: Message: StatusNotificationRequest - connectorStatus Available Message: NotifyEventRequest - eventData[0].trigger Delta - eventData[0].actualValue "Available" - eventData[0].component.name "Connector" - eventData[0].variable.name "AvailabilityState" <p>Post scenario validations: - A message to report the state of a connector has been received for all connectors.</p>

Table 39. Test Case Id: TC_B_23_CS

Test case name	Reset Charging Station - Unavailable persists reset											
Test case Id	TC_B_23_CS											
Use case Id(s)	B11											
Requirement(s)	B11.FR.01, B11.FR.02, B11.FR.03, B11.FR.04, B01.FR.03											
System under test	Charging Station											
Description	This test case covers how the CSMS can request the Charging Station to reset itself by sending a ResetRequest without any ongoing transaction and with the status of Inoperative. This could for example be necessary if the Charging Station is not functioning correctly.											
Purpose	To verify if the Charging Station is able to perform the reset mechanism as described at the OCPP specification.											
Prerequisite(s)	n/a											
Before (Preparations)	<p>Configuration State: N/a</p> <p>Memory State: <i>Unavailable</i> for <Configured connectorId></p> <p>Reusable State(s): N/a</p>											
Main (Test scenario)	<table border="1"> <tr> <td>Charging Station</td> <td>CSMS</td> </tr> <tr> <td>2. The Charging Station responds with a ResetResponse</td> <td>1. The OCTT sends a ResetRequest with type OnIdle</td> </tr> <tr> <td>Note(s): - The Charging Station reboots</td> <td></td> </tr> <tr> <td>3. The Charging Station sends a BootNotificationRequest</td> <td>4. The OCTT responds with a BootNotificationResponse with status Accepted</td> </tr> <tr> <td>5. The Charging Station notifies the CSMS about the current state of all connectors.</td> <td>6. The OCTT responds accordingly.</td> </tr> </table>		Charging Station	CSMS	2. The Charging Station responds with a ResetResponse	1. The OCTT sends a ResetRequest with type OnIdle	Note(s): - The Charging Station reboots		3. The Charging Station sends a BootNotificationRequest	4. The OCTT responds with a BootNotificationResponse with status Accepted	5. The Charging Station notifies the CSMS about the current state of all connectors.	6. The OCTT responds accordingly.
Charging Station	CSMS											
2. The Charging Station responds with a ResetResponse	1. The OCTT sends a ResetRequest with type OnIdle											
Note(s): - The Charging Station reboots												
3. The Charging Station sends a BootNotificationRequest	4. The OCTT responds with a BootNotificationResponse with status Accepted											
5. The Charging Station notifies the CSMS about the current state of all connectors.	6. The OCTT responds accordingly.											

Test case name	Reset Charging Station - Unavailable persists reset
Tool validations	<ul style="list-style-type: none"> * Step 2: Message ResetResponse - status Accepted * Step 3: Message BootNotificationRequest reason RemoteReset * Step 5: For <Configured connectorId>: Message: StatusNotificationRequest - connectorStatus Unavailable Message: NotifyEventRequest - eventData[0].trigger Delta - eventData[0].actualValue "Unavailable" - eventData[0].component.name "Connector" - eventData[0].variable.name "AvailabilityState" For <Other connector(s)>: Message: StatusNotificationRequest - connectorStatus Available Message: NotifyEventRequest - eventData[0].trigger Delta - eventData[0].actualValue "Available" - eventData[0].component.name "Connector" - eventData[0].variable.name "AvailabilityState"
	<p>Post scenario validations:</p> <ul style="list-style-type: none"> - A message to report the state of a connector has been received for all connectors.

Table 40. Test Case Id: TC_B_41_CS

Test case name	Reset Charging Station - With multiple ongoing transactions - OnIdle	
Test case Id	TC_B_41_CS	
Use case Id(s)	B12	
Requirement(s)	B12.FR.01, B12.FR.03, E07.FR.03	
System under test	Charging Station	
Description	<p>This test case covers how the CSMS can remotely request the Charging Station to reset itself by sending a ResetRequest during a transaction. When ResetRequest "OnIdle" is send the charging stations schedules a reboot after all transactions are stopped.</p> <p>This could for example be necessary if the Charging Station is not functioning correctly.</p>	
Purpose	To verify if the Charging Station is able to perform the reset mechanism while there are multiple ongoing transactions as described at the OCPP specification.	
Prerequisite(s)	The Charging Station has more than one EVSE.	
Before (Preparations)	<p>Configuration State: N/a</p> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>EnergyTransferStarted</i> for EVSE.id = 1 State is <i>EnergyTransferStarted</i> for EVSE.id = 2</p>	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with a ResetResponse	1. The OCTT sends a ResetRequest with type OnIdle
	3. Execute Reusable State <i>StopAuthorized</i> for EVSE.id = 1	
	4. Execute Reusable State <i>EVConnectedPostSession</i> for EVSE.id = 1	
	5. Execute Reusable State <i>EVDisconnected</i> for EVSE.id = 1	
	6. Execute Reusable State <i>ParkingBayUnoccupied</i> for EVSE.id = 1	
	7. Execute Reusable State <i>StopAuthorized</i> for EVSE.id = 2	
	8. Execute Reusable State <i>EVConnectedPostSession</i> for EVSE.id = 2	
	Note(s): If TxStopPoint contains one of the following values; Authorized, EnergyTransfer, PowerPathClosed, DataSigned. Then the transaction will have ended at the <i>EVConnectedPostSession</i> state AND the Charging Station will proceed with resetting itself. Proceed to step 10 Else proceed with step 9.	
	9. Execute Reusable State <i>EVDisconnected</i> for EVSE.id = 2	
	Note(s): If TxStopPoint contains the value EVConnected. Then the transaction will have ended at the <i>EVDisconnected</i> state AND the Charging Station will proceed with resetting itself. Proceed to step 11 Else proceed with step 10	
	10. Execute Reusable State <i>ParkingBayUnoccupied</i> for EVSE.id = 2	
	Note(s): The transaction will end at this state, if it was not ended at an earlier state. Proceed to step 11.	
	11. The Charging Station sends a BootNotificationRequest	12. The OCTT responds with a BootNotificationResponse
	13. The Charging Station notifies the CSMS about the current state of all connectors.	14. The OCTT responds accordingly.

Test case name	Reset Charging Station - With multiple ongoing transactions - OnIdle
Tool validations	<ul style="list-style-type: none"> * Step 2: Message ResetResponse - status Scheduled * Step 11: Message BootNotificationRequest - reason ScheduledReset * Step 13: Message: StatusNotificationRequest - If the transaction was stopped at step 3, then connectorStatus Occupied Else connectorStatus Available Message: NotifyEventRequest - If the transaction was stopped at step 3, then eventData[0].actualValue "Occupied" Else eventData[0].actualValue "Available" - eventData[0].trigger Delta - eventData[0].component.name "Connector" - eventData[0].variable.name "AvailabilityState"
	<p>Post scenario validations:</p> <ul style="list-style-type: none"> - A message to report the state of a connector has been received for all connectors.

Table 41. Test Case Id: TC_B_25_CS

Test case name	Reset EVSE - Without ongoing transaction	
Test case Id	TC_B_25_CS	
Use case Id(s)	B11	
Requirement(s)	B11.FR.01, B11.FR.08, B11.FR.10	
System under test	Charging Station	
Description	This test case covers how the CSMS can request the Charging Station to reset an EVSE by sending a ResetRequest without any ongoing transaction. This could for example be necessary if the Charging Station is not functioning correctly.	
Purpose	To verify if the Charging Station is able to perform the reset mechanism as described at the OCPP specification.	
Prerequisite(s)	Individual resetting EVSE supported	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with a ResetResponse	1. The OCTT sends a ResetRequest with type OnIdle and *evsId*<Configured evsId>
	Note(s): - <Configured evsId> reboots	
Tool validations	<ul style="list-style-type: none"> * Step 2: Message ResetResponse - status Accepted 	
	Post scenario validations: - N/a	

Table 42. Test Case Id: TC_B_26_CS

Test case name	Reset EVSE - With Ongoing Transaction - OnIdle	
Test case Id	TC_B_26_CS	
Use case Id(s)	B12	
Requirement(s)	B12.FR.01, B12.FR.07, E07.FR.03	
System under test	Charging Station	
Description	<p>This test case covers how the CSMS can remotely request the Charging Station to reset an EVSE by sending a ResetRequest during a transaction. When ResetRequest "OnIdle" is send the charging stations schedules a reboot after all transactions are stopped.</p> <p>This could for example be necessary if the Charging Station is not functioning correctly.</p>	
Purpose	To verify if the Charging Station is able to perform the reset mechanism as described at the OCPP specification.	
Prerequisite(s)	Individual resetting EVSE supported	
Before (Preparations)	<p>Configuration State: N/a</p> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>EnergyTransferStarted</i></p>	
Main (Test scenario)	<p>Charging Station</p> <p>2. The Charging Station responds with a ResetResponse</p> <p>3. Execute Reusable State <i>StopAuthorized</i></p> <p>4. Execute Reusable State <i>EVConnectedPostSession</i></p> <p>5. Execute Reusable State <i>EVDDisconnected</i></p> <p>6. Execute Reusable State <i>ParkingBayUnoccupied</i></p> <p>7. ChargingStation Reboots</p>	<p>CSMS</p> <p>1. The OCTT sends a ResetRequest with type OnIdle and evselId <Configured evselId></p>
Tool validations	<p>* Step 2: Message ResetResponse - status Scheduled</p> <p>Post scenario validations: N/a</p>	

Table 43. Test Case Id: TC_B_27_CS

Test case name	Reset EVSE - With Ongoing Transaction - Immediate	
Test case Id	TC_B_27_CS	
Use case Id(s)	B12	
Requirement(s)	B12.FR.02, B12.FR.08, E07.FR.03	
System under test	Charging Station	
Description	<p>This test case covers how the CSMS can remotely request the Charging Station to reset an EVSE by sending a ResetRequest during a transaction. When ResetRequest "Immediate" is send the charging stations will try to stop all transactions before rebooting.</p> <p>This could for example be necessary if the Charging Station is not functioning correctly.</p>	
Purpose	To verify if the Charging Station is able to perform the reset mechanism as described at the OCPP specification.	
Prerequisite(s)	Individual resetting EVSE supported	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): State is <i>EnergyTransferStarted</i>	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with a ResetResponse	1. The OCTT sends a ResetRequest with type Immediate and *evseld*<Configured evseld>
	3. The Charging Station sends a TransactionEventRequest	4. The OCTT responds with a TransactionEventResponse
	Note(s): - The EVSE reboots	
Tool validations	<ul style="list-style-type: none"> * Step 2: Message ResetResponse - status Accepted * Step 3: Message TransactionEventRequest - eventType Ended - triggerReason ResetCommand - transactionInfo.chargingState EVConnected - transactionInfo.stoppedReason ImmediateReset 	
	Post scenario validations: - N/a	

Table 44. Test Case Id: TC_B_28_CS

Test case name	Reset EVSE - Not Supported	
Test case Id	TC_B_28_CS	
Use case Id(s)	B11, B12	
Requirement(s)	B11.FR.01, B11.FR.09, B12.FR.01, B12.FR.09	
System under test	Charging Station	
Description	This test case covers how the CSMS can request the Charging Station to reset an EVSE by sending a ResetRequest while it is not supported by the Charging Station.	
Purpose	To verify if the Charging Station is able to perform the reset mechanism as described at the OCPP specification.	
Prerequisite(s)	Charging Station does not support resetting individual EVSE	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): N/a	
Main (Test scenario)	Charging Station 2. The Charging Station responds with a ResetResponse	CSMS 1. The OCTT sends a ResetRequest with type OnIdle and *evseld*<Configured evseld>
Tool validations	* Step 2: Message ResetResponse - status Rejected Post scenario validations: - N/a	

Table 45. Test Case Id: TC_B_29_CS

Test case name	Reset EVSE - With ongoing transaction - Not Supported	
Test case Id	TC_B_29_CS	
Use case Id(s)	B11	
Requirement(s)	B12.FR.01, B12.FR.09	
System under test	Charging Station	
Description	This test case covers how the CSMS can request the Charging Station to reset an EVSE by sending a ResetRequest with ongoing transaction while it is not supported by the Charging Station.	
Purpose	To verify if the Charging Station is able to perform the reset mechanism as described at the OCPP specification.	
Prerequisite(s)	Charging Station does not support resetting individual EVSE	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): State is <i>EnergyTransferStarted</i>	
Main (Test scenario)	Charging Station 2. The Charging Station responds with a ResetResponse	CSMS 1. The OCTT sends a ResetRequest with type OnIdle and evselid <Configured evselid>
Tool validations	* Step 2: Message ResetResponse - status Rejected Post scenario validations: - N/a	

Table 46. Test Case Id: TC_B_43_CS

Test case name	Set new NetworkConnectionProfile - Rejected	
Test case Id	TC_B_43_CS	
Use case Id(s)	B09	
Requirement(s)	B09.FR.02	
System under test	Charging Station	
Description	The CSMS updates the connection details on the Charging Station. For instance in preparation of a migration to a new CSMS.	
Purpose	To verify if the Charging Station is able to reject when the CSMS tries to set a network connection profile containing invalid data.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): N/a	
Main (Test scenario)	Charging Station 2. The Charging Station responds with a SetNetworkProfileResponse	CSMS 1. The OCTT sends a SetNetworkProfileRequest with: <ul style="list-style-type: none"> - configurationSlot is 999 - connectionData.messageTimeout <Configured messageTimeout> - connectionData.ocppCsmsUrl <Configured ocppCsmsUrl> - connectionData.ocppInterface <Configured ocppInterface> - connectionData.ocppVersion OCPP20 - connectionData.securityProfile <Configured securityProfile>
Tool validations	* Step 2: Message SetNetworkProfileResponse - status Rejected Post scenario validations: - N/a	

Table 47. Test Case Id: TC_B_45_CS

Test case name	Migrate to new ConnectionProfile - Success - Same CSMS Root	
Test case Id	TC_B_45_CS	
Use case Id(s)	B09, B10	
Requirement(s)	B09.FR.01,B10.FR.01,B10.FR.04,B10.FR.06	
System under test	Charging Station	
Description	The CSMS updates the connection details on the Charging Station. For instance in preparation of a migration to a new CSMS.	
Purpose	To verify if the Charging Station is able to migrate to another network connection profile slot.	
Prerequisite(s)	At least two configuration slots for networkConnectionProfiles must be supported	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): N/a	
Main (Test scenario)	Charging Station 2. The Charging Station responds with a SetNetworkProfileResponse	CSMS 1. The OCTT sends a SetNetworkProfileRequest with configurationSlot is <i><Configured configurationSlot></i> or <i><Configured configurationSlot></i> depending on which one is already in use - connectionData.messageTimeout <i><Configured messageTimeout></i> - connectionData.ocppCsmsUrl <i><ocppCsmsUrl that is not currently active></i> - connectionData.ocppInterface <i><Configured ocppInterface></i> - connectionData.ocppVersion <i>OCPP20</i> - connectionData.securityProfile <i><Configured securityProfile></i>
	4. The Charging Station responds with a SetVariablesResponse	3. The OCTT sends a SetVariablesRequest with variable.name is "NetworkConfigurationPriority" component.name is "OCPPCommCtrlr" attributeValue is <i><Configured configurationSlot2></i> , <i><Configured configurationSlot></i>
	6. The Charging Station responds with a ResetResponse	5. The OCTT sends a ResetRequest with type <i>OnIdle</i> <u>Note(s):</u> - This step will only be executed when the status <i>RebootRequired</i> is returned at step 4.
	7. Execute Reusable State <i>Booted</i>	<u>Note(s):</u> - The Charging Station connects using the <i><Configured connectionData2></i> .

Test case name	Migrate to new ConnectionProfile - Success - Same CSMS Root
Tool validations	<ul style="list-style-type: none"> * Step 2: Message SetNetworkProfileResponse - status Accepted * Step 4: Message SetVariablesResponse - setVariableResult[0].attributeStatus Accepted OR RebootRequired * Step 6: Message ResetResponse - status Accepted
	Post scenario validations: - N/a

Table 48. Test Case Id: TC_B_46_CS

Test case name	Migrate to new ConnectionProfile - Fallback mechanism - Same CSMS Root	
Test case Id	TC_B_46_CS	
Use case Id(s)	B10	
Requirement(s)	B10.FR.03,B10.FR.04	
System under test	Charging Station	
Description	The CSMS updates the connection details on the Charging Station. For instance in preparation of a migration to a new CSMS.	
Purpose	To verify if the Charging Station is able to use the fallback mechanism when it is unable to connect with the first network connection profile slot.	
Prerequisite(s)	At least two configuration slots for networkConnectionProfiles must be supported	
Before (Preparations)	Configuration State: OCPPCommCtrlr.NetworkProfileConnectionAttempts is 2 Memory State: N/a Reusable State(s): N/a	
Main (Test scenario)	Charging Station 2. The Charging Station responds with a SetNetworkProfileResponse	CSMS 1. The OCTT sends a SetNetworkProfileRequest with configurationSlot is <Configured configurationSlot> or <Configured configurationSlot> depending on which one is already in use - connectionData.messageTimeout <Configured messageTimeout> - connectionData.ocppCsmsUrl <invalid ocppCsmsUrl> - connectionData.ocppInterface <Configured ocppInterface> - connectionData.ocppVersion OCPP20 - connectionData.securityProfile <Configured securityProfile>
	4. The Charging Station responds with a SetVariablesResponse	3. The OCTT sends a SetVariablesRequest with variable.name is "NetworkConfigurationPriority" component.name is "OCPPCommCtrlr" attributeValue is <Configured configurationSlot2>,<Configured configurationSlot>
	6. The Charging Station responds with a ResetResponse	5. The OCTT sends a ResetRequest with type OnIdle <u>Note(s):</u> - This step will only be executed when the status RebootRequired is returned at step 4.
		7. The OCTT will NOT respond to the two connection request from the Charging Station from the first connectionSlot.
		8. The OCTT will accept the connection request from the Charging Station from the second connectionSlot.
	<u>Note(s):</u> Set the <Configured Long Operation Time Out> so that Steps 7 and 8 can be completed in this time period.	
	9. Execute Reusable State Booted <u>Note(s):</u> - The Charging Station connects using the <Configured connectionData>.	

Test case name	Migrate to new ConnectionProfile - Fallback mechanism - Same CSMS Root
Tool validations	<ul style="list-style-type: none"> * Step 2: Message SetNetworkProfileResponse - status Accepted * Step 4: Message SetVariablesResponse - setVariableResult[0].attributeStatus Accepted OR RebootRequired * Step 6: Message ResetResponse - status Accepted <p>Post scenario validations: - N/a</p>

Table 49. Test Case Id: TC_B_47_CS

Test case name	Migrate to new ConnectionProfile - Fallback after NetworkProfileConnectionAttempts per NetworkConfigurationPriority failed - New CSMS Root - New CSMS
Test case Id	TC_B_47_CS
Use case Id(s)	B09,B10,M05
Requirement(s)	B10.FR.07,M05.FR.15,M05.FR.16
System under test	Charging Station
Description	The CSMS updates the connection details on the Charging Station. For instance in preparation of a migration to a new CSMS.
Purpose	To verify if the Charging Station is able to correctly handle the fallback mechanism in the case it fails to connect to the other CSMS.
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station supports AS-2: AdditionalRootCertificateCheck. - Configured (new) CSMS Root certificate 2 must be signed by the configured (old) CSMS Root certificate 2. - At least two configuration slots for networkConnectionProfiles must be supported
Before (Preparations)	<p>Configuration State: OCPPCommCtrlr.NetworkProfileConnectionAttempts is 1</p> <p>Memory State: <i>CertificateInstalled</i> for certificateType CSMSRootCertificate and certificate <Configured (new) CSMS Root certificate 2></p> <p>Reusable State(s): N/a</p>

Test case name	Migrate to new ConnectionProfile - Fallback after NetworkProfileConnectionAttempts per NetworkConfigurationPriority failed - New CSMS Root - New CSMS	
Main (Test scenario)	<p>Charging Station</p> <p>2. The Charging Station responds with a SetNetworkProfileResponse</p>	<p>CSMS</p> <p>1. The OCTT sends a SetNetworkProfileRequest with configurationSlot is <i><Configured configurationSlot></i> or <i><Configured configurationSlot></i> depending on which one is already in use</p> <ul style="list-style-type: none"> - connectionData.messageTimeout <i><Configured messageTimeout2></i> - connectionData.ocppCsmsUrl <i><ocppCsmsUrl that is not currently active></i> - connectionData.ocppInterface <i><Configured ocppInterface2></i> - connectionData.ocppVersion <i>OCPP20</i> - connectionData.securityProfile <i><Configured securityProfile2></i>
	<p>4. The Charging Station responds with a SetVariablesResponse</p>	<p>3. The OCTT sends a SetVariablesRequest with variable.name is "NetworkConfigurationPriority" component.name is "OCPPCommCtrlr" attributeValue is <i><Configured configurationSlot2></i></p>
	<p>6. The Charging Station responds with a ResetResponse</p>	<p>5. The OCTT sends a ResetRequest with type <i>OnIdle</i></p>
	<p>8. During the TLS handshake the Charging Station validates the CSMS certificate.</p> <p><u>Note(s):</u></p> <ul style="list-style-type: none"> - This connection attempt fails, because the Charging Station will use the new CSMS Root certificate to validate the CSMS certificate. 	<p>7. During the TLS handshake the OCTT provides a CSMS certificate which is signed by the <i><Configured old CSMS Root certificate></i></p>
	<p>9. The Charging Station switches back to the previous networkprofile configuration and validates the CSMS certificate, using the (fallback) CSMS Root certificate.</p> <p><u>Note(s):</u></p> <ul style="list-style-type: none"> - This connection attempt succeeds, because the Charging Station will now use the (old) CSMS Root certificate to validate the CSMS certificate. 	
	<p>10. Execute Reusable State <i>Booted</i></p>	
	<p>12. The Charging Station responds with a GetInstalledCertificateIdsResponse</p>	<p>11. The OCTT sends a GetInstalledCertificateIdsRequest with certificateType is <i>CSMSRootCertificate</i></p>
Tool validations	<ul style="list-style-type: none"> * Step 6: Message ResetResponse - status Accepted * Step 12: Message: GetInstalledCertificateIdsResponse - status must be Accepted - certificateHashDataChain must contain an entry with following values: - certificateType is <i>CSMSRootCertificate</i> - certificateHashData contains <i><HashData from configured old CSMS Root certificate></i> <p>Post scenario validations: - N/a</p>	

Table 50. Test Case Id: TC_B_49_CS

Test case name	Migrate to new ConnectionProfile - Fallback after NetworkProfileConnectionAttempts per NetworkConfigurationPriority failed - Same CSMS Root
Test case Id	TC_B_49_CS
Use case Id(s)	B10
Requirement(s)	B10.FR.07
System under test	Charging Station
Description	The CSMS updates the connection details on the Charging Station. For instance in preparation of a migration to a new CSMS.
Purpose	To verify if the Charging Station is able to correctly handle the fallback mechanism in the case it fails to connect to the other CSMS.
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station supports C-47: mechanism implemented & Reconnect after NetworkProfileConnectionAttempts - At least two configuration slots for networkConnectionProfiles must be supported
Before (Preparations)	<p>Configuration State: OCPPCommCtrlr.NetworkProfileConnectionAttempts is 1 OCPPCommCtrlr.RetryBackOffRepeatTimes is 0 OCPPCommCtrlr.RetryBackOffRandomRange is 0 OCPPCommCtrlr.RetryBackOffWaitMinimum is <Configured RetryBackOffWaitMinimum></p> <p>Memory State: N/a</p> <p>Reusable State(s): N/a</p>

Test case name	Migrate to new ConnectionProfile - Fallback after NetworkProfileConnectionAttempts per NetworkConfigurationPriority failed - Same CSMS Root	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with a SetNetworkProfileResponse	<p>1. The OCTT sends a SetNetworkProfileRequest with configurationSlot is <i><Configured configurationSlot></i> or <i><Configured configurationSlot></i> depending on which one is already in use</p> <ul style="list-style-type: none"> - connectionData.messageTimeout <i><Configured messageTimeout></i> - connectionData.ocppCsmsUrl <i><ocppCsmsUrl that is not currently active></i> - connectionData.ocppInterface <i><Configured ocppInterface></i> - connectionData.ocppVersion <i>OCPP20</i> - connectionData.securityProfile <i><Configured securityProfile></i>
	4. The Charging Station responds with a SetVariablesResponse	<p>3. The OCTT sends a SetVariablesRequest with variable.name is "NetworkConfigurationPriority" component.name is "OCPPCommCtrlr" attributeValue is <i><Configured configurationSlot2></i></p>
	6. The Charging Station responds with a ResetResponse	<p>5. The OCTT sends a ResetRequest with type <i>OnIdle</i></p>
	7. The Charging Station tries to connect to the alternative internal OCTT endpoint. Note(s): - Make sure to set the <i><Configured Long Operation Time Out></i> to be the time required for the CS to revert to the previous network profile configuration.	<p>8. The connection attempt is not accepted by the OCTT.</p>
	9. The Charging Station switches back to the previous networkprofile configuration and reconnects to the OCTT.	<p>10. The connection attempt is not accepted by the OCTT.</p>
	11. The Charging Station waits for the duration of the configured RetryBackOffWaitMinimum and reconnects to the OCTT.	<p>12. The connection attempt is accepted by the OCTT.</p>
	13. Execute Reusable State <i>Booted</i>	
Tool validations	<ul style="list-style-type: none"> * Step 6: Message ResetResponse - status Accepted <p>Post scenario validations:</p> <ul style="list-style-type: none"> - N/a 	

Table 51. Test Case Id: TC_B_50_CS

Test case name	Migrate to new ConnectionProfile - Success - New CSMS Root - New CSMS	
Test case Id	TC_B_50_CS	
Use case Id(s)	B10,M05	
Requirement(s)	M05.FR.13	
System under test	Charging Station	
Description	The CSMS updates the connection details on the Charging Station. For instance in preparation of a migration to a new CSMS.	
Purpose	To verify if the Charging Station is able to correctly handle migrating to the new CSMS using a new CSMS Root certificate to validate the server certificate.	
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station supports AS-2: AdditionalRootCertificateCheck. - Configured (new) CSMS Root certificate 2 must be signed by the configured (old) CSMS Root certificate 2. - At least two configuration slots for networkConnectionProfiles must be supported 	
Before (Preparations)	<p>Configuration State: N/a</p> <p>Memory State: <i>CertificateInstalled</i> for certificateType CSMSRootCertificate and certificate <Configured (new) CSMS Root certificate 2></p> <p>Reusable State(s): N/a</p>	
Main (Test scenario)	<p>Charging Station</p> <p>2. The Charging Station responds with a SetNetworkProfileResponse</p> <p>4. The Charging Station responds with a SetVariablesResponse</p> <p>6. The Charging Station responds with a ResetResponse</p> <p>7. The Charging Station connects to the configured alternative internal OCTT endpoint.</p> <p>Note(s): - During the TLS handshake the Charging Station validates and accepts the CSMS certificate, signed by the <Configured (new) CSMS Root certificate 2>.</p> <p>9. Execute Reusable State Booted</p> <p>11. The Charging Station responds with a GetInstalledCertificateIdsResponse</p>	<p>CSMS</p> <p>1. The OCTT sends a SetNetworkProfileRequest with configurationSlot is <Configured configurationSlot> or <Configured configurationSlot> depending on which one is already in use</p> <ul style="list-style-type: none"> - connectionData.messageTimeout <Configured messageTimeout2> - connectionData.ocppCsmsUrl <ocppCsmsUrl that is not currently active> - connectionData.ocppInterface <Configured ocppInterface2> - connectionData.ocppVersion OCPP20 - connectionData.securityProfile <Configured securityProfile2> <p>3. The OCTT sends a SetVariablesRequest with variable.name is "NetworkConfigurationPriority" component.name is "OCPPCommCtrlr" attributeValue is <Configured configurationSlot2></p> <p>5. The OCTT sends a ResetRequest with type OnIdle</p> <p>8. The connection attempt is accepted by the OCTT.</p> <p>10. The OCTT sends a GetInstalledCertificateIdsRequest with certificateType is CSMSRootCertificate</p>

Test case name	Migrate to new ConnectionProfile - Success - New CSMS Root - New CSMS
Tool validations	<ul style="list-style-type: none"> * Step 6: Message ResetResponse - status Accepted * Step 11: Message: GetInstalledCertificateIdsResponse - status must be <i>Accepted</i> - certificateHashDataChain must NOT contain an entry with following values: - certificateType is <i>CSMSRootCertificate</i> - certificateHashData contains <i><HashData from configured old CSMS Root certificate></i> <i>NOTE:</i> The Charging Station dropped the (old) fallback certificate, because it was able to connect using the (new) Root certificate.
	Post scenario validations: <ul style="list-style-type: none"> - N/a

Table 52. Test Case Id: TC_B_51_CS

Test case name	Status change during offline period - > Offline Threshold	
Test case Id	TC_B_51_CS	
Use case Id(s)	B04	
Requirement(s)	B04.FR.01	
System under test	Charging Station	
Description	A Charging Station sends a notification to the CSMS to inform the CSMS about a Connector status change. This can be done in two ways. Via a StatusNotificationRequest or a NotifyEventRequest from the device model. When the Charging Station is connected again to the CSMS after having been offline, and the Charging Station was longer offline than the configured threshold, it will report the status of every connector.	
Purpose	To verify whether the Charging Station reports the status of all connectors after having been offline for longer than the configured threshold with the configuration variable OfflineThreshold .	
Prerequisite(s)	If the Charging Station does not have more than one EVSE, this testcase will be equal to TC_B_52_CS.	
Before (Preparations)	Configuration State: OCPPCommCtrlr.OfflineThreshold is <Configured offlineThreshold> OCPPCommCtrlr.RetryBackOffWaitMinimum is <Configured offlineThreshold> + 2 seconds OCPPCommCtrlr.RetryBackOffRandomRange is 0	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
		1. The OCTT closes the WebSocket connection AND does not accept a reconnect.
	2. Manual Action: Connect the EV and EVSE.	
		3. The OCTT accepts reconnection attempt from the Charging Station, after the configured threshold has been exceeded.
	4. The Charging Station notifies the CSMS about the current state of all connectors.	5. The OCTT responds accordingly.

Test case name	Status change during offline period -> Offline Threshold
Tool validations	<ul style="list-style-type: none"> * Step 4: <p><i>Configured EVSE/Connector:</i></p> <p>Message: StatusNotificationRequest</p> <ul style="list-style-type: none"> - connectorStatus Occupied - evselId <Configured evselId> - connectorId <Configured connectorId> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none"> - eventData[0].trigger Delta - eventData[0].actualValue "Occupied" - eventData[0].component.name "Connector" - eventData[0].variable.name "AvailabilityState" - eventData[0].evse.id <Configured evselId> - eventData[0].connectorId <Configured connectorId> <p><i>All other EVSE/Connector(s):</i></p> <p>Message: StatusNotificationRequest</p> <ul style="list-style-type: none"> - connectorStatus Available <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none"> - eventData[0].trigger Delta - eventData[0].actualValue "Available" - eventData[0].component.name "Connector" - eventData[0].variable.name "AvailabilityState" <p>Post scenario validations:</p> <p>N/a</p>

Table 53. Test Case Id: TC_B_52_CS

Test case name	Status change during offline period - < Offline Threshold											
Test case Id	TC_B_52_CS											
Use case Id(s)	B04											
Requirement(s)	B04.FR.02											
System under test	Charging Station											
Description	A Charging Station sends a notification to the CSMS to inform the CSMS about a Connector status change. This can be done in two ways. Via a StatusNotificationRequest or a NotifyEventRequest from the device model. When the Charging Station is connected again to the CSMS after having been offline, and the Charging Station was shorter offline than the configured threshold, it will report the status of all connector that received a status change.											
Purpose	To verify whether the Charging Station reports the status of connectors that received a status change after having been offline for shorter than the configured threshold with the configuration variable OfflineThreshold .											
Prerequisite(s)	N/a											
Before (Preparations)	<p>Configuration State: OCPPCommCtrlr.OfflineThreshold is <i><Configured offlineThreshold></i> OCPPCommCtrlr.RetryBackOffWaitMinimum is <i><Configured offlineThreshold> - 2 seconds</i> OCPPCommCtrlr.RetryBackOffRandomRange is 0</p> <p>Memory State: N/a</p> <p>Reusable State(s): N/a</p>											
Main (Test scenario)	<table border="1"> <tr> <td>Charging Station</td> <td>CSMS</td> </tr> <tr> <td>1. The OCTT closes the WebSocket connection AND does not accept a reconnect.</td> <td></td> </tr> <tr> <td>2. <u>Manual Action</u>: Connect the EV and EVSE.</td> <td></td> </tr> <tr> <td>3. The OCTT accepts reconnection attempt from the Charging Station.</td> <td></td> </tr> <tr> <td>4. The Charging Station notifies the CSMS about the current state of the configured connector.</td> <td>5. The OCTT responds accordingly.</td> </tr> </table>	Charging Station	CSMS	1. The OCTT closes the WebSocket connection AND does not accept a reconnect.		2. <u>Manual Action</u> : Connect the EV and EVSE.		3. The OCTT accepts reconnection attempt from the Charging Station.		4. The Charging Station notifies the CSMS about the current state of the configured connector.	5. The OCTT responds accordingly.	
Charging Station	CSMS											
1. The OCTT closes the WebSocket connection AND does not accept a reconnect.												
2. <u>Manual Action</u> : Connect the EV and EVSE.												
3. The OCTT accepts reconnection attempt from the Charging Station.												
4. The Charging Station notifies the CSMS about the current state of the configured connector.	5. The OCTT responds accordingly.											
Tool validations	<ul style="list-style-type: none"> * Step 3: <p>Message: StatusNotificationRequest</p> <ul style="list-style-type: none"> - connectorStatus Occupied - evseld <Configured evseld> - connectorId <Configured connectorId> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none"> - eventData[0].trigger Delta - eventData[0].actualValue "Occupied" - eventData[0].component.name "Connector" - eventData[0].variable.name "AvailabilityState" - eventData[0].evse.id <Configured evseld> - eventData[0].connectorId <Configured connectorId> <p>Post scenario validations: N/a</p>											

Table 54. Test Case Id: TC_B_53_CS

Test case name	Get Base Report - Test mandatory DM variables via FullInventory								
Test case Id	TC_B_53_CS								
Use case Id(s)	B07								
Requirement(s)	Chapter Referenced Components and Variables								
System under test	Charging Station								
Description	CSMS requests a FullInventory base report.								
Purpose	To test that Charging Station supports all required DM variables.								
Prerequisite(s)	N/a								
Before (Preparations)	Configuration State: N/a								
	Memory State: N/a								
	Reusable State(s): N/a								
Main (Test scenario)	Charging Station	CSMS							
	2. CS responds with: GetBaseReportResponse with status = Accepted 3. CS sends one or more NotifyReportRequest messages to report all its component/variables.	1. OCTT requests a GetBaseReportRequest with: reportBase = <i>FullInventory</i> and requestId = <Generated requestId> 4. OCTT responds with a NotifyReportResponse for each NotifyReportRequest							
Tool validations	* Step 2: Message: GetBaseReportResponse with: - status = Accepted - statusInfo is absent or statusInfo = "NoError"								
	* step 3: Message: NotifyReportRequest with: - requestId = <Generated requestId> - generatedAt = <time of generation at charging station> - seqNo = 0								
	While tbc = true		Message: NotifyReportRequest - seqNo is incremented by one						
	Post scenario validations: OCTT checks that at least the following variables are reported:								

Component	Variable	Instance	Mutability	DataType	Other
Required Core components/variables					
AlignedDataCtrlr	Interval		ReadWrite	integer	unit="s"
AlignedDataCtrlr	Measurands		ReadWrite	MemberList	maxLimit>0, valuesList=subset(<supported measurands>)
AlignedDataCtrlr	TxEndedInterval		ReadWrite	integer	unit="s"
AlignedDataCtrlr	TxEndedMeasurands		ReadWrite	MemberList	maxLimit>0, valuesList=subset(<supported measurands>)
AuthCtrlr	AuthorizeRemoteStart		ReadOnly, ReadWrite	boolean	
AuthCtrlr	LocalAuthorizeOffline		ReadWrite	boolean	
AuthCtrlr	LocalPreAuthorize		ReadWrite	boolean	

ChargingStation	AvailabilityState		ReadOnly	OptionList	valuesList="Available, Occupied, Reserved, Unavailable, Faulted"
ChargingStation	Available		ReadOnly	boolean	
ChargingStation	SupplyPhases		ReadOnly	integer	
ClockCtrlr	DateTime		ReadOnly	dateTime	
ClockCtrlr	TimeSource		ReadWrite	SequenceList	valuesList=subset(Heartbeat, NTP, GPS, RealTimeClock, MobileNetwork, RadioTimeTransmitter)
Connector	AvailabilityState		ReadOnly	OptionList	valuesList="Available, Occupied, Reserved, Unavailable, Faulted"
Connector	Available		ReadOnly	boolean	
Connector	ConnectorType		ReadOnly	string	
Connector	SupplyPhases		ReadOnly	integer	
DeviceDataCtrlr	BytesPerMessage	GetReport	ReadOnly	integer	
DeviceDataCtrlr	BytesPerMessage	GetVariables	ReadOnly	integer	
DeviceDataCtrlr	BytesPerMessage	SetVariables	ReadOnly	integer	
DeviceDataCtrlr	ItemsPerMessage	GetReport	ReadOnly	integer	
DeviceDataCtrlr	ItemsPerMessage	GetVariables	ReadOnly	integer	
DeviceDataCtrlr	ItemsPerMessage	SetVariable	ReadOnly	integer	
EVSE	AvailabilityState		ReadOnly	OptionList	valuesList="Available, Occupied, Reserved, Unavailable, Faulted"
EVSE	Available		ReadOnly	boolean	
EVSE	Power		ReadOnly	decimal	unit=oneOf(W, kW), maxLimit>0
EVSE	SupplyPhases		ReadOnly	integer	
OCPPCommCtrlr	FileTransferProtocols		ReadOnly	MemberList	valuesList=subset(FTP, FTPS, HTTP, HTTPS, SFTP)
OCPPCommCtrlr	MessageTimeout	Default	ReadOnly	integer	unit="s"
OCPPCommCtrlr	MessageAttemptInterval	TransactionEvent	ReadWrite	integer	unit="s"
OCPPCommCtrlr	MessageAttempts	TransactionEvent	ReadWrite	integer	
OCPPCommCtrlr	NetworkConfigurationPriority		ReadWrite	SequenceList	valuesList=subset(1, 2, 3, 4, 5)
OCPPCommCtrlr	NetworkProfileConnectionAttempts		ReadWrite	integer	
OCPPCommCtrlr	OfflineThreshold		ReadWrite	integer	unit="s"
OCPPCommCtrlr	ResetRetries		ReadWrite	integer	
OCPPCommCtrlr	UnlockOnEVSideDisconnect		ReadWrite, ReadOnly	boolean	
SampledDataCtrlr	TxEndedInterval		ReadWrite	integer	unit="s"
SampledDataCtrlr	TxEndedMeasurands		ReadWrite	MemberList	maxLimit>0, valuesList=subset(<supported measurands>)

SampledDataCtrlr	TxStartedMeasurands		ReadWrite	MemberList	maxLimit>0, valuesList=subset(<supported measurands>)
SampledDataCtrlr	TxUpdatedInterval		ReadWrite	integer	unit="s"
SampledDataCtrlr	TxUpdatedMeasurands		ReadWrite	MemberList	maxLimit>0, valuesList=subset(<supported measurands>)
SecurityCtrlr	CertificateEntries		ReadOnly	integer	maxLimit>0
SecurityCtrlr	OrganizationName		ReadWrite	string	
SecurityCtrlr	SecurityProfile		ReadOnly	integer	
TxCtrlr	EVConnectionTimeOut		ReadWrite	integer	unit="s"
TxCtrlr	StopTxOnEVSideDisconnect		ReadWrite, ReadOnly	boolean	
TxCtrlr	StopTxOnInvalidId		ReadWrite	boolean	
TxCtrlr	TxStartPoint		ReadWrite, ReadOnly	MemberList	valuesList=subset(ParkingBayOccupancy, EVConnected, Authorized, PowerPathClosed, EnergyTransfer, DataSigned)
TxCtrlr	TxStopPoint		ReadWrite, ReadOnly	MemberList	valuesList=subset(ParkingBayOccupancy, EVConnected, Authorized, PowerPathClosed, EnergyTransfer, DataSigned)

Required DisplayMessageCtrlr variables

DisplayMessageCtrlr	DisplayMessages		ReadOnly	integer	
DisplayMessageCtrlr	SupportedFormats		ReadOnly	MemberList	valuesList=subset(ASCII, HTML, URI, UTF8)
DisplayMessageCtrlr	SupportedPriorities		ReadOnly	MemberList	valuesList=subset(AlwaysFront, InFront, NormalCycle)

Required ISO15118Ctrlr variables

ISO15118Ctrlr	ContractValidationOffline		ReadWrite	boolean	
---------------	---------------------------	--	-----------	---------	--

Required LocalAuthListCtrlr variables

LocalAuthListCtrlr	BytesPerMessage		ReadOnly	integer	
LocalAuthListCtrlr	Entries		ReadOnly	integer	maxLimit>0
LocalAuthListCtrlr	ItemsPerMessage		ReadOnly	integer	

Required MonitoringCtrlr variables

MonitoringCtrlr	BytesPerMessage	SetVariableMonitoring	ReadOnly	integer	
-----------------	-----------------	-----------------------	----------	---------	--

Required SmartChargingCtrlr variables

SmartChargingCtrlr	Entries	ChargingProfiles	ReadOnly	integer	maxLimit>0
SmartChargingCtrlr	LimitChangeSignificance		ReadWrite	decimal	unit="Percent"
SmartChargingCtrlr	PeriodsPerSchedule		ReadOnly	integer	
SmartChargingCtrlr	ProfileStackLevel		ReadOnly	integer	
SmartChargingCtrlr	RateUnit		ReadOnly	MemberList	valuesList="A, W"

Required TariffCostCtrlr variables

TariffCostCtrlr	Currency			string	
TariffCostCtrlr	TariffFallbackMessage		ReadWrite	string	maxLimit=255
TariffCostCtrlr	TotalCostFallbackMessage		ReadWrite	string	maxLimit=255

Table 55. Test Case Id: TC_B_57_CS

Test case name	Network Reconnection - After connection loss	
Test case Id	TC_B_57_CS	
Use case Id(s)	Part 4 section 5.3. Reconnecting	
Requirement(s)	Described at section 5.3.	
System under test	Charging Station	
Description	When the connection is lost, the Charging Station SHALL try to reconnect. When reconnecting, the Charging Station SHALL use an increasing back-off time until it has successfully reconnected.	
Purpose	To verify if the Charging Station is able to reconnect to the CSMS using the described OCPP reconnecting mechanism from part 4.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: OCPPCommCtrlr.RetryBackOffRepeatTimes is 2 OCPPCommCtrlr.RetryBackOffRandomRange is 0 OCPPCommCtrlr.RetryBackOffWaitMinimum is <Configured RetryBackOffWaitMinimum> Memory State: N/a Reusable State(s): N/a	
Main (Test scenario)	Charging Station 1. The OCTT closes the websocket connection. 2. The Charging Station waits for the duration of the configured RetryBackOffWaitMinimum and reconnects to the OCTT. 4. The OCTT closes the websocket connection. 5. The Charging Station waits for the duration of the configured RetryBackOffWaitMinimum and reconnects to the OCTT. 7. The Charging Station waits for the duration of the at step 3 doubled RetryBackOffWaitMinimum and reconnects to the OCTT.	CSMS 3. The connection attempt is accepted by the OCTT. 6. The connection attempt is not accepted by the OCTT. 8. The connection attempt is accepted by the OCTT.
Tool validations	<ul style="list-style-type: none"> * Step 2: - The reconnection time is at least the configured RetryBackOffWaitMinimum. * Step 7: - The reconnection time is at least 3 times the reconnection time from step 2. <p>Post scenario validations: - N/a</p>	

2.4. C Authorization

Table 56. Test Case Id: TC_C_02_CS

Test case name	Local start transaction - Authorization Invalid/Unknown	
Test case Id	TC_C_02_CS	
Use case Id(s)	C01 OR C04 OR C06	
Requirement(s)	C01.FR.02 OR C06.FR.02	
System under test	Charging Station	
Description	When a Charging Station needs to charge an EV, it needs to authorize the EV Driver first at the CSMS before the charging can be started or stopped.	
Purpose	To verify whether the Charging Station is able to handle receiving an invalid idToken.	
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station supports at least one authorization method described at the following Use cases; C01, C04, C06. - The Charging Station does NOT have a cable lock, which prevents the EV driver to connect the EV and EVSE before authorization. 	
Before (Preparations)	<p>Configuration State: AuthCtrlr.AuthEnabled is <i>true</i> (If implemented AND ReadWrite) AuthCtrlr.DisableRemoteAuthorization is <i>false</i> (If implemented)</p> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>EVConnectedPreSession</i></p>	
Main (Test scenario)	<p>Charging Station</p> <p><u>Manual Action:</u> Present <i>idToken</i>.</p> <p>1. The Charging Station sends an AuthorizeRequest</p>	<p>CSMS</p> <p>2. The OCTT responds with an AuthorizeResponse with idTokenInfo.status <i>Invalid</i></p>
	<p>Note(s):</p> <ul style="list-style-type: none"> - The Charging Station <i>SHALL NOT</i> send a <i>TransactionEventRequest</i> message after the <i>AuthorizeRequest</i> from step 1. - The OCTT waits <Configured message timeout> seconds, before ending the testcase. 	
Tool validations	<p>* Step 1: Message: AuthorizeRequest</p> <ul style="list-style-type: none"> - idToken.idToken <Configured invalid_idtoken_idtoken> - idToken.type <Configured invalid_idtoken_type> <p>Post scenario validations: N/a</p>	

Table 57. Test Case Id: TC_C_05_CS

Test case name	Local start transaction - Authorization invalid - Cable lock						
Test case Id	TC_C_05_CS						
Use case Id(s)	C01 OR C04 OR C06						
Requirement(s)	C01.FR.02 OR C06.FR.02						
System under test	Charging Station						
Description	When a Charging Station needs to charge an EV, it needs to authorize the EV Driver first before the charging can be started or stopped.						
Purpose	To verify whether a Charging Station with a cable lock, which prevents the EV driver to connect the EV and EVSE before authorization, is able to handle receiving an invalid idToken.						
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station has a cable lock, which prevents the EV driver to connect the EV and EVSE before authorization. - The Charging Station supports at least one authorization method described at the following Use cases; C01, C04, C06. - The Charging Station does NOT have the following configuration; TxStartPoint ReadOnly AND value Authorized is NOT set. 						
Before (Preparations)	<p>Configuration State: AuthCtrlr.AuthEnabled is <i>true</i> (If implemented AND ReadWrite) AuthCtrlr.DisableRemoteAuthorization is <i>false</i> (If implemented)</p> <p>Memory State: N/a</p> <p>Reusable State(s): N/a</p>						
Main (Test scenario)	<table border="1" style="width: 100%;"> <tr> <td>Charging Station</td> <td>CSMS</td> </tr> <tr> <td colspan="2"><u>Manual Action:</u> Present <i>idToken</i>.</td> </tr> <tr> <td colspan="2"> <p>1. The Charging Station sends an AuthorizeRequest</p> <p>2. The OCTT responds with an AuthorizeResponse with idTokenInfo.status <i>Invalid</i></p> </td> </tr> </table> <p>Note(s):</p> <ul style="list-style-type: none"> - The Charging Station <i>SHALL NOT</i> send a <i>TransactionEventRequest</i> message after the <i>AuthorizeRequest</i> from step 1. - The OCTT waits <Configured message timeout> seconds, before ending the testcase. 	Charging Station	CSMS	<u>Manual Action:</u> Present <i>idToken</i> .		<p>1. The Charging Station sends an AuthorizeRequest</p> <p>2. The OCTT responds with an AuthorizeResponse with idTokenInfo.status <i>Invalid</i></p>	
Charging Station	CSMS						
<u>Manual Action:</u> Present <i>idToken</i> .							
<p>1. The Charging Station sends an AuthorizeRequest</p> <p>2. The OCTT responds with an AuthorizeResponse with idTokenInfo.status <i>Invalid</i></p>							
Tool validations	<p>* Step 1: Message: AuthorizeRequest - idToken.idToken <Configured invalid_idtoken_idtoken> - idToken.type <Configured invalid_idtoken_type></p> <p>Post scenario validations: N/a</p>						

Table 58. Test Case Id: TC_C_04_CS

Test case name	Local Stop Transaction - Different idToken	
Test case Id	TC_C_04_CS	
Use case Id(s)	C01, C04, E07	
Requirement(s)	N/a	
System under test	Charging Station	
Description	The EV Driver is able to stop an ongoing transaction, by locally presenting his IdToken.	
Purpose	To verify whether the Charging Station does not stop the charging session when a different idToken is presented, than the one used to start the transaction.	
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station supports at least one authorization method described at the following Use cases; C01, C04. - The Charging Station does NOT use one idToken reader for multiple EVSE. 	
Before (Preparations)	<p>Configuration State: AuthCtrlr.AuthEnabled is <i>true</i> (If implemented AND ReadWrite) AuthCtrlr.DisableRemoteAuthorization is <i>false</i> (If implemented)</p> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>EnergyTransferStarted</i></p>	
Main (Test scenario)	<p>Charging Station</p> <p><u>Manual Action:</u> Present a different idToken than used to start the transaction.</p> <p>1. The Charging Station sends an AuthorizeRequest</p>	<p>CSMS</p> <p>2. The OCTT responds with an AuthorizeResponse with idTokenInfo.status Accepted</p>
	<p>Note(s):</p> <ul style="list-style-type: none"> - The Charging Station SHALL NOT send an AuthorizeRequest AND/OR a TransactionEventRequest message after receiving an idToken that is different, than the one used to start the transaction. - The OCTT waits <Configured message timeout> seconds, before ending the testcase. 	
Tool validations	<p>N/a</p> <p>Post scenario validations: N/a</p>	

Table 59. Test Case Id: TC_C_06_CS

Test case name	Local start transaction - Authorization Blocked	
Test case Id	TC_C_06_CS	
Use case Id(s)	C01	
Requirement(s)	C01.FR.02	
System under test	Charging Station	
Description	When a Charging Station needs to charge an EV, it needs to authorize the EV Driver first at the CSMS before the charging can be started or stopped.	
Purpose	To verify whether the Charging Station is able to handle receiving an Blocked idToken.	
Prerequisite(s)	The Charging Station does NOT have a cable lock, which prevents the EV driver to connect the EV and EVSE before authorization.	
Before (Preparations)	<p>Configuration State: AuthCtrlr.AuthEnabled is <i>true</i> (If implemented AND ReadWrite) AuthCtrlr.DisableRemoteAuthorization is <i>false</i> (If implemented)</p> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>EVConnectedPreSession</i></p>	
Main (Test scenario)	<p>Charging Station</p> <p><u>Manual Action:</u> Present <i>idToken</i>.</p> <p>1. The Charging Station sends an AuthorizeRequest</p>	<p>CSMS</p> <p>2. The OCTT responds with an AuthorizeResponse with idTokenInfo.status <i>Blocked</i></p>
	<p>Note(s):</p> <ul style="list-style-type: none"> - The Charging Station SHALL NOT send a <i>TransactionEventRequest</i> message after the <i>AuthorizeRequest</i> from step 7. - The OCTT waits <Configured message timeout> seconds, before ending the testcase. 	
Tool validations	<ul style="list-style-type: none"> * Step 1: Message: AuthorizeRequest - idToken.idToken <Configured blocked_idtoken_idtoken> - idToken.type <Configured blocked_idtoken_type> <p>Post scenario validations: N/a</p>	

Table 60. Test Case Id: TC_C_07_CS

Test case name	Local start transaction - Authorization Expired	
Test case Id	TC_C_07_CS	
Use case Id(s)	C01	
Requirement(s)	C01.FR.02	
System under test	Charging Station	
Description	When a Charging Station needs to charge an EV, it needs to authorize the EV Driver first at the CSMS before the charging can be started or stopped.	
Purpose	To verify whether the Charging Station is able to handle receiving an Expired idToken.	
Prerequisite(s)	The Charging Station does NOT have a cable lock, which prevents the EV driver to connect the EV and EVSE before authorization.	
Before (Preparations)	<p>Configuration State: AuthCtrlIr.AuthEnabled is <i>true</i> (If implemented AND ReadWrite) AuthCtrlIr.DisableRemoteAuthorization is <i>false</i> (If implemented)</p> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>EVConnectedPreSession</i></p>	
Main (Test scenario)	<p>Charging Station</p> <p><u>Manual Action:</u> Present <i>idToken</i>.</p> <p>1. The Charging Station sends an AuthorizeRequest</p>	<p>CSMS</p> <p>2. The OCTT responds with an AuthorizeResponse with idTokenInfo.status <i>Expired</i></p>
	<p>Note(s):</p> <ul style="list-style-type: none"> - The Charging Station SHALL NOT send a <i>TransactionEventRequest</i> message after the <i>AuthorizeRequest</i> from step 7. - The OCTT waits <Configured message timeout> seconds, before ending the testcase. 	
Tool validations	<ul style="list-style-type: none"> * Step 1: Message: AuthorizeRequest - idToken.idToken <Configured expired_idtoken_idtoken> - idToken.type <Configured expired_idtoken_type> <p>Post scenario validations: N/a</p>	

Table 61. Test Case Id: TC_C_08_CS

Test case name	Authorization through authorization cache - Accepted	
Test case Id	TC_C_08_CS	
Use case Id(s)	C12	
Requirement(s)	C12_FR_02, C12_FR_04	
System under test	Charging Station	
Description	This test case describes how the EV Driver is authorized to start a transaction while the Charging Station uses Cached IdToken. This enables the EV Driver to Online start a transaction by using the Authorization Cache in which the Charging Station can respond faster, as no AuthorizeRequest is being sent.	
Purpose	To verify if the Charging Station is able to Authorize an idToken which has status "Accepted" in its cache according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	AuthCacheCtrlr.AuthCacheAvailable is implemented with value true	
Before (Preparations)	<p>Configuration State: AuthCacheCtrlr.AuthCacheEnabled is <i>true</i> (If implemented) AuthCtrlr.LocalPreAuthorize is <i>true</i> (If implemented)</p> <p>Memory State: <i>IdTokenCached</i> for <Configured valid IdToken fields></p> <p>Reusable State(s): State is <i>EVConnectedPreSession</i></p>	
Main (Test scenario)	Charging Station	CSMS
	<p>1. Execute Reusable State <i>Authorized</i></p> <p><u>Note(s):</u> Present valid idToken which is already configured in the Authorization Cache</p> <p>2. Execute Reusable State <i>EnergyTransferStarted</i></p>	
Tool validations	<p>N/a</p> <p>Post scenario validations: - Energy transfer is started</p>	

Table 62. Test Case Id: TC_C_09_CS

Test case name	Authorization through authorization cache - Invalid & Not Accepted	
Test case Id	TC_C_09_CS	
Use case Id(s)	C12	
Requirement(s)	C12_FR_05, C10_FR_03	
System under test	Charging Station	
Description	This test case describes how the EV Driver is authorized to start a transaction while the Charging Station uses Cached IdToken. This enables the EV Driver to Online start a transaction by using the Authorization Cache in which the Charging Station can respond faster, as no AuthorizeRequest is being sent.	
Purpose	To verify if the Charging Station is able to Authorize an idToken which has status "Invalid" in its cache according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	AuthCacheCtrlr.AuthCacheAvailable is implemented with value true	
Before (Preparations)	<p>Configuration State: AuthCacheCtrlr.AuthCacheEnabled is <i>true</i> (If implemented) AuthCtrlr.LocalPreAuthorize is <i>true</i> (If implemented) AuthCacheCtrlrDisablePostAuthorize is <i>false</i> (If implemented)</p> <p>Memory State: <i>IdTokenCached</i> for <Configured invalid IdToken fields></p> <p>Reusable State(s): State is <i>EVConnectedPreSession</i></p>	
Main (Test scenario)	<p>Charging Station</p> <p><u>Manual Action:</u> Present Invalid idToken which is already configured in the Authorization Cache</p>	<p>CSMS</p> <p>1. The Charging Station sends a AuthorizeRequest</p> <p>2. The OCTT responds with a AuthorizeResponse with idTokenInfo.status <i>Invalid</i></p>
Tool validations	<p>* Step 1: Message AuthorizeRequest - idToken.idToken <Configured invalid_idtoken_idtoken> - idToken.type <Configured invalid_idtoken_type></p> <p>Post scenario validations: - N/a</p>	

Table 63. Test Case Id: TC_C_10_CS

Test case name	Authorization through authorization cache - Blocked	
Test case Id	TC_C_10_CS	
Use case Id(s)	C12	
Requirement(s)	C12_FR_05, C10_FR_03	
System under test	Charging Station	
Description	This test case describes how the EV Driver is authorized to start a transaction while the Charging Station uses Cached IdToken. This enables the EV Driver to Online start a transaction by using the Authorization Cache in which the Charging Station can respond faster, as no AuthorizeRequest is being sent.	
Purpose	To verify if the Charging Station is able to Authorize an idToken which has status "Blocked" in its cache according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	AuthCacheCtrlr.AuthCacheAvailable is implemented with value true	
Before (Preparations)	<p>Configuration State: AuthCacheCtrlr.AuthCacheEnabled is <i>true</i> (If implemented) AuthCtrlr.LocalPreAuthorize is <i>true</i> (If implemented) AuthCacheCtrlrDisablePostAuthorize is <i>false</i> (If implemented)</p> <p>Memory State: <i>IdTokenCached</i> for <Configured blocked IdToken fields></p> <p>Reusable State(s): State is <i>EVConnectedPreSession</i></p>	
Main (Test scenario)	<p>Charging Station</p> <p><u>Manual Action:</u> Present Blocked idToken which is already configured in the Authorization Cache</p>	<p>CSMS</p> <p>1. The Charging Station sends a AuthorizeRequest</p> <p>2. The OCTT responds with a AuthorizedResponse with idTokenInfo.status Blocked</p>
Tool validations	<p>* Step 1: Message AuthorizeRequest - idToken.idToken <Configured blocked_idtoken_idtoken> - idToken.type <Configured blocked_idtoken_type></p> <p>Post scenario validations: - N/a</p>	

Table 64. Test Case Id: TC_C_11_CS

Test case name	Authorization through authorization cache - Expired	
Test case Id	TC_C_11_CS	
Use case Id(s)	C12	
Requirement(s)	C12_FR_05, C10_FR_03	
System under test	Charging Station	
Description	This test case describes how the EV Driver is authorized to start a transaction while the Charging Station uses Cached IdToken. This enables the EV Driver to Online start a transaction by using the Authorization Cache in which the Charging Station can respond faster, as no AuthorizeRequest is being sent.	
Purpose	To verify if the Charging Station is able to Authorize an idToken which has status "Expired" in its cache according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	AuthCacheCtrlr.AuthCacheAvailable is implemented with value true	
Before (Preparations)	<p>Configuration State: AuthCacheCtrlr.AuthCacheEnabled is <i>true</i> (If implemented) AuthCtrlr.LocalPreAuthorize is <i>true</i> (If implemented) AuthCacheCtrlrDisablePostAuthorize is <i>false</i> (If implemented)</p> <p>Memory State: <i>IdTokenCached</i> for <Configured expired IdToken fields></p> <p>Reusable State(s): State is <i>EVConnectedPreSession</i></p>	
Main (Test scenario)	<p>Charging Station</p> <p><u>Manual Action:</u> Present Expired idToken which is already configured in the Authorization Cache</p>	<p>CSMS</p> <p>1. The Charging Station sends a AuthorizeRequest</p> <p>2. The OCTT responds with a TransactionEventResponse with idTokenInfo.status <i>Expired</i></p>
Tool validations	<p>* Step 1: Message AuthorizeRequest - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type></p> <p>Post scenario validations: - N/a</p>	

Table 65. Test Case Id: TC_C_12_CS

Test case name	Authorization through authorization cache - Invalid & Accepted	
Test case Id	TC_C_12_CS	
Use case Id(s)	C12	
Requirement(s)	C12_FR_05, C10_FR_03	
System under test	Charging Station	
Description	This test case describes how the EV Driver is authorized to start a transaction while the Charging Station uses Cached IdToken. This enables the EV Driver to Online start a transaction by using the Authorization Cache in which the Charging Station can respond faster, as no AuthorizeRequest is being sent.	
Purpose	To verify if the Charging Station is able to Authorize an idToken which has status "Invalid" in its cache, but the CSMS has status "Valid", according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	AuthCacheCtrlr.AuthCacheAvailable is implemented with value true	
Before (Preparations)	<p>Configuration State: AuthCacheCtrlr.AuthCacheEnabled is <i>true</i> (If implemented) AuthCtrlr.LocalPreAuthorize is <i>true</i> (If implemented) AuthCacheCtrlrDisablePostAuthorize is <i>false</i> (If implemented)</p> <p>Memory State: <i>IdTokenCached</i> for <Configured invalid IdToken fields></p> <p>Reusable State(s): State is <i>EVConnectedPreSession</i></p>	
Main (Test scenario)	<p>Charging Station</p> <p><u>Manual Action:</u> Present Invalid idToken which is already configured in the Authorization Cache</p> <p>1. The Charging Station sends an AuthorizeRequest</p> <p>2. The OCTT responds with an AuthorizeResponse with idTokenInfo.status Accepted</p> <p>3. The Charging Station sends a TransactionEventRequest <u>Note(s):</u> -This step is optional.</p> <p>4. The OCTT responds with a TransactionEventResponse</p> <p>5. Execute Reusable State <i>EnergyTransferStarted</i></p>	<p>CSMS</p>
Tool validations	<ul style="list-style-type: none"> * Step 1: Message AuthorizeRequest <ul style="list-style-type: none"> - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type> Post scenario validations: <ul style="list-style-type: none"> - Energy transfer is started 	

Table 66. Test Case Id: TC_C_13_CS

Test case name	Authorization through authorization cache - Accepted but cable not connected yet.											
Test case Id	TC_C_13_CS											
Use case Id(s)	C12											
Requirement(s)	C12_FR_02, C12_FR_04											
System under test	Charging Station											
Description	This test case describes how the EV Driver is authorized to start a transaction while the Charging Station uses Cached IdToken. This enables the EV Driver to Online start a transaction by using the Authorization Cache in which the Charging Station can respond faster, as no AuthorizeRequest is being sent.											
Purpose	To verify if the Charging Station is able to Authorize an idToken which has status "Accepted" in its cache but the cable is not connected yet according to the mechanism as described in the OCPP specification.											
Prerequisite(s)	AuthCacheCtrlr.AuthCacheAvailable is implemented with value true											
Before (Preparations)	<p>Configuration State: AuthCacheCtrlr.AuthCacheEnabled is <i>true</i> (If implemented) AuthCtrlr.LocalPreAuthorize is <i>true</i> (If implemented)</p> <p>Memory State: <i>IdTokenCached</i> for <Configured valid IdToken fields> (If implemented)</p> <p>Reusable State(s): If applicable, State is <i>ParkingBayOccupied</i></p>											
Main (Test scenario)	<table border="1"> <tr> <td>Charging Station</td> <td>CSMS</td> </tr> <tr> <td colspan="2"><u>Manual Action: Present valid idToken which is already configured in the Authorization Cache</u></td> </tr> <tr> <td> 1. The Charging Station sends a TransactionEventRequest <u>Note(s):</u> - This step needs to be executed when TxStartPoint contains Authorized OR the transaction already started. So in the case TxStartPoint contains ParkingBayOccupancy. - As long as the cable is not plugged in the energy transfer will not start. </td> <td> 2. The OCTT responds with a TransactionEventResponse with idTokenInfo.status Accepted </td> </tr> <tr> <td colspan="2">3. Execute Reusable State <i>EVConnectedPreSession</i></td> </tr> <tr> <td colspan="2">4. Execute Reusable State <i>EnergyTransferStarted</i></td> </tr> </table>	Charging Station	CSMS	<u>Manual Action: Present valid idToken which is already configured in the Authorization Cache</u>		1. The Charging Station sends a TransactionEventRequest <u>Note(s):</u> - This step needs to be executed when TxStartPoint contains Authorized OR the transaction already started. So in the case TxStartPoint contains ParkingBayOccupancy. - As long as the cable is not plugged in the energy transfer will not start.	2. The OCTT responds with a TransactionEventResponse with idTokenInfo.status Accepted	3. Execute Reusable State <i>EVConnectedPreSession</i>		4. Execute Reusable State <i>EnergyTransferStarted</i>		
Charging Station	CSMS											
<u>Manual Action: Present valid idToken which is already configured in the Authorization Cache</u>												
1. The Charging Station sends a TransactionEventRequest <u>Note(s):</u> - This step needs to be executed when TxStartPoint contains Authorized OR the transaction already started. So in the case TxStartPoint contains ParkingBayOccupancy. - As long as the cable is not plugged in the energy transfer will not start.	2. The OCTT responds with a TransactionEventResponse with idTokenInfo.status Accepted											
3. Execute Reusable State <i>EVConnectedPreSession</i>												
4. Execute Reusable State <i>EnergyTransferStarted</i>												
Tool validations	<ul style="list-style-type: none"> * Step 1: Message TransactionEventRequest - triggerReason Authorized - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type> if transaction was already started - eventType Updated else - eventType Started <p>Post scenario validations:</p> <ul style="list-style-type: none"> - Energy transfer is started 											

Table 67. Test Case Id: TC_C_15_CS

Test case name	Authorization through authorization cache - StopTxOnInvalidId = false, MaxEnergyOnInvalidId > 0																	
Test case Id	TC_C_15_CS																	
Use case Id(s)	C12																	
Requirement(s)	C12_FR_02, C12_FR_04																	
System under test	Charging Station																	
Description	This test case describes how the EV Driver is authorized to start a transaction while the Charging Station uses Cached IdToken. This enables the EV Driver to Online start a transaction by using the Authorization Cache in which the Charging Station can respond faster, as no AuthorizeRequest is being sent.																	
Purpose	To verify if the Charging Station is able to respond correctly to an idToken which has status "Accepted" in its cache but is "Invalid" in the CSMS with certain values of StopTxOnInvalidId and MaxEnergyOnInvalidId according to the mechanism as described in the OCPP specification.																	
Prerequisite(s)	<ul style="list-style-type: none"> - AuthCacheCtrlr.AuthCacheAvailable is implemented with value true - The Charging Station has MaxEnergyOnInvalidId implemented - At least one of the following must be supported; Local auth list, auth cache, StartTxUnknownIds. 																	
Before (Preparations)	<p>Configuration State:</p> <p>AuthCacheCtrlr.AuthCacheEnabled is <i>true</i> (If implemented)</p> <p>AuthCtrlr.LocalPreAuthorize is <i>true</i> (If implemented)</p> <p>AuthCtrlr.LocalAuthorizeOffline is <i>true</i></p> <p>OfflineTxForUnknownIdEnabled is <i>true</i> (If implemented)</p> <p>StopTxOnInvalidId is <i>false</i></p> <p>MaxEnergyOnInvalidId is <i>500</i></p> <p>OfflineThreshold is <i><Configured RetryBackOffWaitMinimum_duration> + 60.0</i></p> <p>RetryBackOffWaitMinimum is <i><Configured RetryBackOffWaitMinimum_duration></i></p> <p>RetryBackOffRandomRange is <i>0</i></p> <p>Note: <i><Configured RetryBackOffWaitMinimum_duration should be long enough to execute manual tasks></i></p> <p>Memory State: <i>IdTokenCached</i> for <i><Configured valid IdToken fields></i> (If implemented) A known valid idToken is configured in the Local auth list (if implemented) and auth cache (if implemented)</p> <p>Reusable State(s): State is <i>EVConnectedPreSession</i></p>																	
Main (Test scenario)	<table border="1"> <thead> <tr> <th>Charging Station</th> <th>CSMS</th> </tr> </thead> <tbody> <tr> <td>1. The OCTT closes the WebSocket connection AND does not accept a reconnect.</td> <td></td> </tr> <tr> <td>Manual Action: Present valid idToken which is already configured in the Authorization Cache</td> <td></td> </tr> <tr> <td>Note(s): The OCTT will wait for _<Configured Transaction Duration> seconds_</td> <td></td> </tr> <tr> <td>2. The OCTT accepts reconnection attempt from the Charging Station.</td> <td></td> </tr> <tr> <td>Note(s): - The Charging Station will empty its Transaction message queue. This will contain one or more TransactionEventRequest messages</td> <td></td> </tr> <tr> <td>3. The Charging Station sends a TransactionEventRequest</td> <td>4. The OCTT responds with a TransactionEventResponse with idTokenInfo.status Invalid (if idToken is not omitted)</td> </tr> <tr> <td>5. The Charging Station sends a TransactionEventRequest with triggerReason Deauthorized</td> <td>6. The OCTT responds with a TransactionEventResponse</td> </tr> </tbody> </table>		Charging Station	CSMS	1. The OCTT closes the WebSocket connection AND does not accept a reconnect.		Manual Action: Present valid idToken which is already configured in the Authorization Cache		Note(s): The OCTT will wait for _<Configured Transaction Duration> seconds_		2. The OCTT accepts reconnection attempt from the Charging Station.		Note(s): - The Charging Station will empty its Transaction message queue. This will contain one or more TransactionEventRequest messages		3. The Charging Station sends a TransactionEventRequest	4. The OCTT responds with a TransactionEventResponse with idTokenInfo.status Invalid (if idToken is not omitted)	5. The Charging Station sends a TransactionEventRequest with triggerReason Deauthorized	6. The OCTT responds with a TransactionEventResponse
Charging Station	CSMS																	
1. The OCTT closes the WebSocket connection AND does not accept a reconnect.																		
Manual Action: Present valid idToken which is already configured in the Authorization Cache																		
Note(s): The OCTT will wait for _<Configured Transaction Duration> seconds_																		
2. The OCTT accepts reconnection attempt from the Charging Station.																		
Note(s): - The Charging Station will empty its Transaction message queue. This will contain one or more TransactionEventRequest messages																		
3. The Charging Station sends a TransactionEventRequest	4. The OCTT responds with a TransactionEventResponse with idTokenInfo.status Invalid (if idToken is not omitted)																	
5. The Charging Station sends a TransactionEventRequest with triggerReason Deauthorized	6. The OCTT responds with a TransactionEventResponse																	

Test case name	Authorization through authorization cache - StopTxOnInvalidId = false, MaxEnergyOnInvalidId > 0
Tool validations	<ul style="list-style-type: none"> * Step 2: <p>Message TransactionEventRequest</p> <p>A message with (optional):</p> <ul style="list-style-type: none"> - triggerReason Authorized - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type> - offline True <p>A message with:</p> <ul style="list-style-type: none"> - triggerReason ChargingStateChanged - offline True <p>No message with:</p> <ul style="list-style-type: none"> - triggerReason Deauthorized or - transactionInfo.chargingState SuspendedEVSE
	<p>Post scenario validations:</p> <ul style="list-style-type: none"> - Energy transfer is started but only MaxEnergyOnInvalidId amount of energy is delivered

Table 68. Test Case Id: TC_C_16_CS

Test case name	Authorization through authorization cache - StopTxOnInvalidId = true																			
Test case Id	TC_C_16_CS																			
Use case Id(s)	C12																			
Requirement(s)	C12_FR_02, C12_FR_04																			
System under test	Charging Station																			
Description	This test case describes how the EV Driver is authorized to start a transaction while the Charging Station uses Cached IdToken. This enables the EV Driver to Online start a transaction by using the Authorization Cache in which the Charging Station can respond faster, as no AuthorizeRequest is being sent.																			
Purpose	To verify if the Charging Station is able to respond correctly to an idToken which has status "Accepted" in its cache but is "Invalid" in the CSMS when StopTxOnInvalidId is true according to the mechanism as described in the OCPP specification.																			
Prerequisite(s)	<ul style="list-style-type: none"> - AuthCacheCtrlr.AuthCacheAvailable is implemented with value true - At least one of the following must be supported; Local auth list, auth cache, StartTxUnknownIds. 																			
Before (Preparations)	<p>Configuration State:</p> <p>AuthCacheEnabled is true (If implemented)</p> <p>LocalPreAuthorize is true (If implemented)</p> <p>LocalAuthorizeOffline is true</p> <p>StopTxOnInvalidId is true</p> <p>MaxEnergyOnInvalidId is 0</p> <p>Memory State:</p> <p>IdTokenCached for <Configured valid IdToken fields> (If implemented)</p> <p>A known valid idToken is configured in the Local auth list (if implemented) and auth cache (if implemented)</p> <p>Reusable State(s):</p> <p>State is EVConnectedPreSession</p>																			
Main (Test scenario)	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left; padding: 2px;">Charging Station</th> <th style="text-align: left; padding: 2px;">CSMS</th> </tr> </thead> <tbody> <tr> <td style="padding: 2px;">1. The OCTT closes the WebSocket connection AND does not accept a reconnect.</td> <td style="padding: 2px;"></td> </tr> <tr> <td colspan="2" style="padding: 2px;">Manual Action: Present valid idToken which is already configured in the Authorization Cache</td> </tr> <tr> <td colspan="2" style="padding: 2px;">Note(s): The OCTT will wait for 5 seconds</td> </tr> <tr> <td colspan="2" style="padding: 2px;">2. The OCTT accepts reconnection attempt from the Charging Station.</td> </tr> <tr> <td colspan="2" style="padding: 2px;">Note(s):</td> </tr> <tr> <td colspan="2" style="padding: 2px;"> <ul style="list-style-type: none"> - The Charging Station will empty its Transaction message queue. This will contain one or more TransactionEventRequest messages </td> </tr> <tr> <td style="padding: 2px; vertical-align: top;"> 3. The Charging Station sends a TransactionEventRequest </td><td style="padding: 2px; vertical-align: top;"> 4. The OCTT responds with a TransactionEventResponse with idTokenInfo.status Invalid (if idToken is not omitted) </td> </tr> <tr> <td style="padding: 2px; vertical-align: top;"> 5. The Charging Station sends a TransactionEventRequest with triggerReason Deauthorized </td><td style="padding: 2px; vertical-align: top;"> 6. The OCTT responds with a TransactionEventResponse </td> </tr> </tbody> </table>		Charging Station	CSMS	1. The OCTT closes the WebSocket connection AND does not accept a reconnect.		Manual Action: Present valid idToken which is already configured in the Authorization Cache		Note(s): The OCTT will wait for 5 seconds		2. The OCTT accepts reconnection attempt from the Charging Station.		Note(s):		<ul style="list-style-type: none"> - The Charging Station will empty its Transaction message queue. This will contain one or more TransactionEventRequest messages 		3. The Charging Station sends a TransactionEventRequest	4. The OCTT responds with a TransactionEventResponse with idTokenInfo.status Invalid (if idToken is not omitted)	5. The Charging Station sends a TransactionEventRequest with triggerReason Deauthorized	6. The OCTT responds with a TransactionEventResponse
Charging Station	CSMS																			
1. The OCTT closes the WebSocket connection AND does not accept a reconnect.																				
Manual Action: Present valid idToken which is already configured in the Authorization Cache																				
Note(s): The OCTT will wait for 5 seconds																				
2. The OCTT accepts reconnection attempt from the Charging Station.																				
Note(s):																				
<ul style="list-style-type: none"> - The Charging Station will empty its Transaction message queue. This will contain one or more TransactionEventRequest messages 																				
3. The Charging Station sends a TransactionEventRequest	4. The OCTT responds with a TransactionEventResponse with idTokenInfo.status Invalid (if idToken is not omitted)																			
5. The Charging Station sends a TransactionEventRequest with triggerReason Deauthorized	6. The OCTT responds with a TransactionEventResponse																			
Tool validations	<ul style="list-style-type: none"> * Step 3: Message TransactionEventRequest A message with (optional): <ul style="list-style-type: none"> - triggerReason Authorized - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type> - offline True A message with: <ul style="list-style-type: none"> - triggerReason ChargingStateChanged - offline True A message with: <ul style="list-style-type: none"> - triggerReason Deauthorized <p>Post scenario validations:</p> <ul style="list-style-type: none"> - Energyflow stops on receiving status invalid 																			

Table 69. Test Case Id: TC_C_17_CS

Test case name	Authorization through authorization cache - StopTxOnInvalidId = false																			
Test case Id	TC_C_17_CS																			
Use case Id(s)	C12																			
Requirement(s)	C12_FR_02, C12_FR_04																			
System under test	Charging Station																			
Description	This test case describes how the EV Driver is authorized to start a transaction while the Charging Station uses Cached IdToken. This enables the EV Driver to Online start a transaction by using the Authorization Cache in which the Charging Station can respond faster, as no AuthorizeRequest is being sent.																			
Purpose	To verify if the Charging Station is able to respond correctly to an idToken which has status "Accepted" in its cache but is "Invalid" in the CSMS when StopTxOnInvalidId is false according to the mechanism as described in the OCPP specification.																			
Prerequisite(s)	<ul style="list-style-type: none"> - AuthCacheCtrlr.AuthCacheAvailable is implemented with value true - At least one of the following must be supported; Local auth list, auth cache, StartTxUnknownIds. 																			
Before (Preparations)	<p>Configuration State:</p> <p>AuthCacheEnabled is <i>true</i> (If implemented)</p> <p>LocalPreAuthorize is <i>true</i> (If implemented)</p> <p>OfflineTxForUnknownIdEnabled is <i>true</i> (If implemented)</p> <p>StopTxOnInvalidId is <i>false</i></p> <p>MaxEnergyOnInvalidId is <i>0</i></p> <p>Memory State:</p> <p><i>IdTokenCached</i> for <Configured valid IdToken fields> (If implemented) A known valid idToken is configured in the Local auth list (if implemented) and auth cache (if implemented)</p> <p>Reusable State(s):</p> <p>State is <i>EVConnectedPreSession</i></p>																			
Main (Test scenario)	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left; padding: 2px;">Charging Station</th> <th style="text-align: left; padding: 2px;">CSMS</th> </tr> </thead> <tbody> <tr> <td colspan="2" style="padding: 2px;">1. The OCTT closes the WebSocket connection AND does not accept a reconnect.</td> </tr> <tr> <td colspan="2" style="padding: 2px;"><u>Manual Action:</u> Present valid idToken which is already configured in the Authorization Cache</td> </tr> <tr> <td colspan="2" style="padding: 2px;"><u>Note(s):</u> The OCTT will wait for 5 seconds</td> </tr> <tr> <td colspan="2" style="padding: 2px;">2. The OCTT accepts reconnection attempt from the Charging Station.</td> </tr> <tr> <td colspan="2" style="padding: 2px;"><u>Note(s):</u></td> </tr> <tr> <td colspan="2" style="padding: 2px;">- The Charging Station will empty its Transaction message queue. This will contain one or more TransactionEventRequest messages</td> </tr> <tr> <td style="vertical-align: top; padding: 2px;">3. The Charging Station sends a TransactionEventRequest</td> <td style="vertical-align: top; padding: 2px;">4. The OCTT responds with a TransactionEventResponse with idTokenInfo.status Invalid (if idToken is not omitted)</td> </tr> <tr> <td style="vertical-align: top; padding: 2px;">5. The Charging Station sends a TransactionEventRequest with triggerReason SuspendedEVSE</td> <td style="vertical-align: top; padding: 2px;">6. The OCTT responds with a TransactionEventResponse</td> </tr> </tbody> </table>		Charging Station	CSMS	1. The OCTT closes the WebSocket connection AND does not accept a reconnect.		<u>Manual Action:</u> Present valid idToken which is already configured in the Authorization Cache		<u>Note(s):</u> The OCTT will wait for 5 seconds		2. The OCTT accepts reconnection attempt from the Charging Station.		<u>Note(s):</u>		- The Charging Station will empty its Transaction message queue. This will contain one or more TransactionEventRequest messages		3. The Charging Station sends a TransactionEventRequest	4. The OCTT responds with a TransactionEventResponse with idTokenInfo.status Invalid (if idToken is not omitted)	5. The Charging Station sends a TransactionEventRequest with triggerReason SuspendedEVSE	6. The OCTT responds with a TransactionEventResponse
Charging Station	CSMS																			
1. The OCTT closes the WebSocket connection AND does not accept a reconnect.																				
<u>Manual Action:</u> Present valid idToken which is already configured in the Authorization Cache																				
<u>Note(s):</u> The OCTT will wait for 5 seconds																				
2. The OCTT accepts reconnection attempt from the Charging Station.																				
<u>Note(s):</u>																				
- The Charging Station will empty its Transaction message queue. This will contain one or more TransactionEventRequest messages																				
3. The Charging Station sends a TransactionEventRequest	4. The OCTT responds with a TransactionEventResponse with idTokenInfo.status Invalid (if idToken is not omitted)																			
5. The Charging Station sends a TransactionEventRequest with triggerReason SuspendedEVSE	6. The OCTT responds with a TransactionEventResponse																			
Tool validations	<ul style="list-style-type: none"> * Step 2: Message TransactionEventRequest A message with: <ul style="list-style-type: none"> - triggerReason Authorized - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type> - offline True A message with: <ul style="list-style-type: none"> - transactionInfo.chargingState SuspendedEVSE No message with: - triggerReason SuspendedEVSE <p>Post scenario validations:</p> <ul style="list-style-type: none"> - Energyflow stops on receiving status invalid 																			

Table 70. Test Case Id: TC_C_18_CS

Test case name	Authorization through authorization cache - StopTxOnInvalidId = true, MaxEnergyOnInvalidId > 0																			
Test case Id	TC_C_18_CS																			
Use case Id(s)	C12																			
Requirement(s)	C12_FR_02, C12_FR_04																			
System under test	Charging Station																			
Description	This test case describes how the EV Driver is authorized to start a transaction while the Charging Station uses Cached IdToken. This enables the EV Driver to Online start a transaction by using the Authorization Cache in which the Charging Station can respond faster, as no AuthorizeRequest is being sent.																			
Purpose	To verify if the Charging Station is able to respond correctly to an idToken which has status "Accepted" in its cache but is "Invalid" in the CSMS when StopTxOnInvalidId is true and MaxEnergyOnInvalidId > 0 according to the mechanism as described in the OCPP specification.																			
Prerequisite(s)	<ul style="list-style-type: none"> - AuthCacheCtrlr.AuthCacheAvailable is implemented with value true - The Charging Station has MaxEnergyOnInvalidId implemented. - At least one of the following must be supported; Local auth list, auth cache, StartTxUnknownIds. 																			
Before (Preparations)	<p>Configuration State:</p> <p>AuthCacheEnabled is true (If implemented)</p> <p>LocalPreAuthorize is true (If implemented)</p> <p>LocalAuthorizeOffline is true</p> <p>OfflineTxForUnknownIdEnabled is true (If implemented)</p> <p>StopTxOnInvalidId is true</p> <p>MaxEnergyOnInvalidId is 500</p> <p>OfflineThreshold is <Configured RetryBackOffWaitMinimum_duration> + 60.0</p> <p>RetryBackOffWaitMinimum is <Configured RetryBackOffWaitMinimum_duration></p> <p>RetryBackOffRandomRange is 0</p> <p>Note: <i><Configured RetryBackOffWaitMinimum_duration should be long enough to execute manual tasks></i></p> <p>Memory State:</p> <p>IdTokenCached for <Configured valid IdToken fields> (If implemented)</p> <p>A known valid idToken is configured in the Local auth list (if implemented) and auth cache (if implemented)</p> <p>Reusable State(s): State is EVConnectedPreSession</p>																			
Main (Test scenario)	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left; padding-bottom: 5px;">Charging Station</th> <th style="text-align: left; padding-bottom: 5px;">CSMS</th> </tr> </thead> <tbody> <tr> <td colspan="2">1. The OCTT closes the WebSocket connection AND does not accept a reconnect.</td> </tr> <tr> <td colspan="2"><u>Manual Action:</u> Present valid idToken which is already configured in the Authorization Cache</td> </tr> <tr> <td colspan="2"><u>Note(s):</u> The OCTT will wait for <Configured Transaction Duration> seconds</td> </tr> <tr> <td colspan="2">2. The OCTT accepts reconnection attempt from the Charging Station.</td> </tr> <tr> <td colspan="2"><u>Note(s):</u></td> </tr> <tr> <td colspan="2"> <ul style="list-style-type: none"> - The Charging Station will empty its Transaction message queue. This will contain one or more TransactionEventRequest messages </td> </tr> <tr> <td style="vertical-align: top; padding-right: 10px;"> 3. The Charging Station sends a TransactionEventRequest </td><td style="padding-left: 10px;"> 4. The OCTT responds with a TransactionEventResponse with idTokenInfo.status Invalid (if idToken is not omitted) </td></tr> <tr> <td style="vertical-align: top; padding-right: 10px;"> 5. The Charging Station sends a TransactionEventRequest with triggerReason Deauthorized </td><td style="padding-left: 10px;"> 6. The OCTT responds with a TransactionEventResponse </td></tr> </tbody> </table>		Charging Station	CSMS	1. The OCTT closes the WebSocket connection AND does not accept a reconnect.		<u>Manual Action:</u> Present valid idToken which is already configured in the Authorization Cache		<u>Note(s):</u> The OCTT will wait for <Configured Transaction Duration> seconds		2. The OCTT accepts reconnection attempt from the Charging Station.		<u>Note(s):</u>		<ul style="list-style-type: none"> - The Charging Station will empty its Transaction message queue. This will contain one or more TransactionEventRequest messages 		3. The Charging Station sends a TransactionEventRequest	4. The OCTT responds with a TransactionEventResponse with idTokenInfo.status Invalid (if idToken is not omitted)	5. The Charging Station sends a TransactionEventRequest with triggerReason Deauthorized	6. The OCTT responds with a TransactionEventResponse
Charging Station	CSMS																			
1. The OCTT closes the WebSocket connection AND does not accept a reconnect.																				
<u>Manual Action:</u> Present valid idToken which is already configured in the Authorization Cache																				
<u>Note(s):</u> The OCTT will wait for <Configured Transaction Duration> seconds																				
2. The OCTT accepts reconnection attempt from the Charging Station.																				
<u>Note(s):</u>																				
<ul style="list-style-type: none"> - The Charging Station will empty its Transaction message queue. This will contain one or more TransactionEventRequest messages 																				
3. The Charging Station sends a TransactionEventRequest	4. The OCTT responds with a TransactionEventResponse with idTokenInfo.status Invalid (if idToken is not omitted)																			
5. The Charging Station sends a TransactionEventRequest with triggerReason Deauthorized	6. The OCTT responds with a TransactionEventResponse																			

Test case name	Authorization through authorization cache - StopTxOnInvalidId = true, MaxEnergyOnInvalidId > 0
Tool validations	<ul style="list-style-type: none"> * Step 3: Message TransactionEventRequest A message with (optional): <ul style="list-style-type: none"> - triggerReason Authorized - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type> - offline True A message with: <ul style="list-style-type: none"> - triggerReason ChargingStateChanged - offline True * Step 5: A message with: <ul style="list-style-type: none"> - triggerReason Deauthorized - offline False
	<p>Post scenario validations:</p> <ul style="list-style-type: none"> - Energyflow stops on receiving status invalid

Table 71. Test Case Id: TC_C_57_CS

Test case name	Authorization through authorization cache - AuthCacheDisablePostAuthorize	
Test case Id	TC_C_57_CS	
Use case Id(s)	C12	
Requirement(s)	C12_FR_05, C10_FR_03	
System under test	Charging Station	
Description	This test case describes how the EV Driver can be authorized to start a transaction by using Cached IdTokens. This enables the EV Driver to start a transaction while the Charging Station is online by using the Authorization Cache in which case the Charging Station can respond faster, since no AuthorizeRequest is being sent. For an IdToken that does not have status "Accepted" the Charging Station will still send an AuthorizeRequest to check against the most recent status from CSMS. However, when the setting AuthCacheDisablePostAuthorize is set to true, then the Charging Station will not do this.	
Purpose	To verify that the Charging Station will not send an AuthorizeRequest for an IdToken in the Authorization Cache that is not "Accepted", when AuthCacheDisablePostAuthorize is set to true.	
Prerequisite(s)	<ul style="list-style-type: none"> - AuthCacheCtrlr.Available is implemented with value <i>true</i> - AuthCacheCtrlr.DisablePostAuthorize is implemented 	
Before (Preparations)	<p>Configuration State: AuthCacheCtrlr.Enabled is <i>true</i> (If implemented) AuthCtrlr.LocalPreAuthorize is <i>true</i> (If implemented) AuthCacheCtrlr.DisablePostAuthorize is <i>true</i></p> <p>Memory State: <i>IdTokenCached</i> for <Configured invalid IdToken fields></p> <p>Reusable State(s): State is <i>EVConnectedPreSession</i></p>	
Main (Test scenario)	Charging Station <u>Manual Action: Present Invalid idToken which is already configured in the Authorization Cache</u> 1. The Charging Station does NOT send a AuthorizeRequest	CSMS
Tool validations	<ul style="list-style-type: none"> * Step 1: Check that Charging Station does NOT send an AuthorizeRequest and authorization is refused. <p>Post scenario validations:</p> <ul style="list-style-type: none"> - N/a 	

Table 72. Test Case Id: TC_C_32_CS

Test case name	Store Authorization Data in the Authorization Cache - Persistent over reboot	
Test case Id	TC_C_32_CS	
Use case Id(s)	C10	
Requirement(s)	C10_FR_02	
System under test	Charging Station	
Description	This test case covers how the Charging Station autonomously stores a record of previously presented identifiers that have been successfully authorized by the CSMS in the Authorization Cache. (Successfully meaning: a response received on a message containing an IdToken)	
Purpose	To verify if the Charging Station is able to store the identifiers persistent over reboot according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station supports the Authorization Cache feature - Authorization cache is stored in the non-volatile memory. 	
Before (Preparations)	<p>Configuration State: AuthCacheEnabled is <i>true</i> (If implemented) LocalPreAuthorize is <i>true</i></p> <p>Memory State: <i>IdTokenCached</i> for <Configured valid IdToken fields></p> <p>Reusable State(s): N/a</p>	
Main (Test scenario)	<p>Charging Station</p> <p>1. Execute Reusable State <i>Booted</i></p> <p>Manual Action: Present valid <i>idToken</i> which is already configured in the Authorization Cache</p> <p>2. The Charging Station sends a TransactionEventRequest</p> <p>Note(s): - This step needs to be executed when TxStartPoint contains Authorized OR the transaction already started. So in the case TxStartPoint contains ParkingBayOccupancy. - As long as the cable is not plugged in the energy transfer will not start.</p>	<p>CSMS</p> <p>3. The OCTT responds with a TransactionEventResponse with idTokenInfo.status Accepted</p> <p>4. Execute Reusable State <i>EVConnectedPreSession</i></p> <p>5. Execute Reusable State <i>EnergyTransferStarted</i></p>
Tool validations	<ul style="list-style-type: none"> * Step 2: Message TransactionEventRequest - triggerReason Authorized - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type> <p>Post scenario validations: - N/a</p>	

Table 73. Test Case Id: TC_C_33_CS

Test case name	Store Authorization Data in the Authorization Cache - Update on AuthorizeResponse	
Test case Id	TC_C_33_CS	
Use case Id(s)	C10	
Requirement(s)	C10.FR.04, C12.FR.06	
System under test	Charging Station	
Description	This test case covers how the Charging Station autonomously stores a record of previously presented identifiers that have been successfully authorized by the CSMS in the Authorization Cache. (Successfully meaning: a response received on a message containing an IdToken)	
Purpose	To verify if the Charging Station is able to store the identifiers correctly upon an AuthorizeResponse according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	- AuthCacheCtrlr.AuthCacheAvailable is implemented with value true	
Before (Preparations)	<p>Configuration State: AuthCacheEnabled is <i>true</i> (If implemented) LocalPreAuthorize is <i>true</i> LocalAuthListEnabled is <i>true</i></p> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>EVConnectedPreSession</i></p>	
Main (Test scenario)	<p>Charging Station</p> <p><u>Manual Action:</u> Present valid idToken</p> <p>1. The Charging Station sends an AuthorizeRequest</p> <p>3. The Charging Station sends a TransactionEventRequest <u>Note(s):</u> - This step needs to be executed when TxStartPoint contains <i>ParkingBayOccupancy</i>, <i>EVConnected</i>, <i>Authorized</i>, or <i>PowerPathClosed</i></p> <p>5. Execute Reusable State <i>EnergyTransferStarted</i></p> <p>6. Execute Reusable State <i>EVConnectedPostSession</i></p> <p>7. Execute Reusable State <i>EVDDisconnected</i></p> <p>8. Execute Reusable State <i>ParkingBayUnoccupied</i></p> <p>9. Execute Reusable State <i>ParkingBayOccupied</i></p> <p>10. Execute Reusable State <i>EVConnectedPreSession</i> <u>Manual Action:</u> Present same valid idToken</p> <p>12. The Charging Station sends a TransactionEventRequest <u>Note(s):</u> - This step needs to be executed when TxStartPoint contains <i>Authorized</i> OR the transaction already started. So in the case TxStartPoint contains <i>ParkingBayOccupancy</i>.</p>	<p>CSMS</p> <p>2. The OCTT responds with an AuthorizeResponse with idTokenInfo.status Accepted</p> <p>4. The OCTT responds with a TransactionEventResponse with idTokenInfo.status Accepted</p> <p>13. The OCTT responds with a TransactionEventResponse with idTokenInfo.status Invalid</p>

Table 74. Test Case Id: TC_C_34_CS

Test case name	Store Authorization Data in the Authorization Cache - Update on TransactionResponse	
Test case Id	TC_C_34_CS	
Use case Id(s)	C10	
Requirement(s)	C10.FR.05, C12.FR.06	
System under test	Charging Station	
Description	This test case covers how the Charging Station autonomously stores a record of previously presented identifiers that have been successfully authorized by the CSMS in the Authorization Cache. (Successfully meaning: a response received on a message containing an IdToken)	
Purpose	To verify if the Charging Station is able to store the identifiers correctly upon an TransactionResponse according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	- AuthCacheCtrlr.AuthCacheAvailable is implemented with value true	
Before (Preparations)	<p>Configuration State: AuthCacheEnabled is <i>true</i> (If implemented) LocalPreAuthorize is <i>true</i> LocalAuthListEnabled is <i>true</i></p> <p>Memory State: <i>IdTokenCached</i> for <Configured valid IdToken fields></p> <p>Reusable State(s): State is <i>EVConnectedPreSession</i></p>	
Main (Test scenario)	<p>Charging Station</p> <p><u>Manual Action:</u> Present valid <i>idToken</i></p> <p>1. The Charging Station sends a TransactionEventRequest</p>	<p>CSMS</p> <p>2. The OCTT responds with a TransactionEventResponse with idTokenInfo.status <i>Invalid</i></p>
	<p>3. Execute Reusable State <i>EVDisconnected</i></p> <p>4. Execute Reusable State <i>ParkingBayUnoccupied</i></p> <p>5. Execute Reusable State <i>ParkingBayOccupied</i></p> <p>6. Execute Reusable State <i>EVConnectedPreSession</i></p> <p><u>Manual Action:</u> Present same valid <i>idToken</i></p>	
	<p>7. The Charging Station sends an AuthorizeRequest</p>	<p>8. The OCTT responds with an AuthorizeResponse with idTokenInfo.status <i>Invalid</i></p>
Tool validations	<ul style="list-style-type: none"> * Step 1: Message TransactionEventRequest <ul style="list-style-type: none"> - triggerReason <i>Authorized</i> - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type> if transaction was already started <ul style="list-style-type: none"> - eventType <i>Updated</i> else <ul style="list-style-type: none"> - eventType <i>Started</i> * Step 7: Message AuthorizeRequest <ul style="list-style-type: none"> - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type> <p>Post scenario validations:</p> <ul style="list-style-type: none"> - N/a 	

Table 75. Test Case Id: TC_C_36_CS

Test case name	Store Authorization Data in the Authorization Cache - AuthCacheCtrlr.LocalPreAuthorize = false	
Test case Id	TC_C_36_CS	
Use case Id(s)	C10	
Requirement(s)	C10_FR_11	
System under test	Charging Station	
Description	This test case covers how the Charging Station autonomously stores a record of previously presented identifiers that have been successfully authorized by the CSMS in the Authorization Cache. (Successfully meaning: a response received on a message containing an IdToken)	
Purpose	To verify if the Charging Station is able to ignore the Authorization Cache feature when LocalPreAuthorize is set to false according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	- AuthCacheCtrlr.AuthCacheAvailable is implemented	
Before (Preparations)	<p>Configuration State: AuthCacheEnabled is <i>true</i> LocalPreAuthorize is <i>false</i></p> <p>Memory State: <i>IdTokenCached</i> for <Configured valid IdToken fields></p> <p>Reusable State(s): N/a</p>	
Main (Test scenario)	<p>Charging Station</p> <p><u>Manual Action:</u> Present valid idToken which is configured in the Authorization Cache</p> <p>1. The Charging Station sends an AuthorizeRequest</p>	<p>CSMS</p> <p>2. The OCTT responds with an AuthorizeResponse with idTokenInfo.status <i>Invalid</i></p>
Tool validations	<p>* Step 1: Message AuthorizeRequest</p> <ul style="list-style-type: none"> - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type> <p>Post scenario validations:</p> <ul style="list-style-type: none"> - N/a 	

Table 76. Test Case Id: TC_C_37_CS

Test case name	Clear Authorization Data in Authorization Cache - Accepted	
Test case Id	TC_C_37_CS	
Use case Id(s)	C11	
Requirement(s)	C11.FR_01, C11.FR.02, C11.FR.03	
System under test	Charging Station	
Description	This test case covers how the Charging Station autonomously stores a record of previously presented identifiers that have been successfully authorized by the CSMS in the Authorization Cache. (Successfully meaning: a response received on a message containing an IdToken)	
Purpose	To verify if the Charging Station is able to clear all identifiers from the Authorization Cache according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	- AuthCacheCtrlr.AuthCacheAvailable is implemented with value true	
Before (Preparations)	<p>Configuration State: AuthCacheEnabled is true (If implemented)</p> <p>Memory State: <i>IdTokenCached</i> for <Configured valid IdToken fields></p> <p>Reusable State(s): N/a</p>	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with a ClearCacheResponse	1. The OCTT sends a ClearCacheRequest
	3. Execute Reusable State <i>ParkingBayOccupied</i>	
	4. Execute Reusable State <i>EVConnectedPreSession</i>	
	<u>Manual Action:</u> Present valid idToken which was configured in the Authorization Cache	
	5. The Charging Station sends an AuthorizeRequest	6. The OCTT responds with an AuthorizeResponse with idTokenInfo.status Accepted
	7. The Charging Station sends an TransactionEventRequest	8. The OCTT responds with an TransactionEventResponse with triggerReason Authorized
	9. Execute Reusable State <i>EnergyTransferStarted</i>	
Tool validations	<ul style="list-style-type: none"> * Step 2: Message ClearCacheResponse - status Accepted * Step 5: Message AuthorizeRequest - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type> <p>Post scenario validations: - N/a</p>	

Table 77. Test Case Id: TC_C_38_CS

Test case name	Clear Authorization Data in Authorization Cache - Rejected	
Test case Id	TC_C_38_CS	
Use case Id(s)	C11	
Requirement(s)	C11.FR_01, C11.FR.02, C11.FR.04	
System under test	Charging Station	
Description	This test case covers how the Charging Station autonomously stores a record of previously presented identifiers that have been successfully authorized by the CSMS in the Authorization Cache. (Successfully meaning: a response received on a message containing an IdToken)	
Purpose	To verify if the Charging Station is able to correctly respond on a request from the CSMS to clear all identifiers from the Authorization Cache while the feature is disabled according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	<ul style="list-style-type: none"> - AuthCacheCtrlr.AuthCacheAvailable is implemented with value true - AuthCacheCtrlr.LocalPreAuthorize is implemented 	
Before (Preparations)	<p>Configuration State: AuthCacheEnabled is <i>false</i> (If implemented)</p> <p>Memory State: <i>IdTokenCached</i> for <Configured valid IdToken fields></p> <p>Reusable State(s): N/a</p>	
Main (Test scenario)	<p>Charging Station</p> <p>2. The Charging Station responds with a ClearCacheResponse</p>	<p>CSMS</p> <p>1. The OCTT sends a ClearCacheRequest</p>
Tool validations	<p>* Step 2: Message ClearCacheResponse - status <i>Rejected</i></p> <p>Post scenario validations: - N/a</p>	

Table 78. Test Case Id: TC_C_39_CS

Test case name	Authorization by GroupId - Success	
Test case Id	TC_C_39_CS	
Use case Id(s)	C09	
Requirement(s)	C09_FR_02, C09_FR_03, C09_FR_05	
System under test	Charging Station	
Description	This test case covers how a Charging Station can authorize an action for an EV Driver based on GroupId information. This could for example be used if 2 people regularly use the same EV: they can use their own IdToken (e.g. RFID card), and can deauthorize transactions that were started with the other idToken (with the same GroupId).	
Purpose	To verify if the Charging Station is able to correctly handle the Authorization of idTokens with the same GroupId according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	- N/a	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): State is <i>EVConnectedPreSession</i>	
Main (Test scenario)	Charging Station <u>Manual Action:</u> Present valid idToken with <Configured GroupId>	CSMS
	1. The Charging Station sends an AuthorizeRequest 3. The Charging Station sends a TransactionEventRequest <u>Note(s):</u> - This step needs to be executed when TxStartPoint contains Authorized OR the transaction already started. So in the case TxStartPoint contains ParkingBayOccupancy	2. The OCTT responds with an AuthorizeResponse with - idTokenInfo.status Accepted - idTokenInfo.groupIdToken.idToken <Configured groupIdToken> 4. The OCTT responds with a TransactionEventResponse with - idTokenInfo.status Accepted - idTokenInfo.groupIdToken.idToken <Configured groupIdToken>
	5. Execute Reusable State <i>EnergyTransferStarted</i> <u>Manual Action:</u> Present other valid idToken with <Configured GroupId>	
	6. The Charging Station sends an AuthorizeRequest	7. The OCTT responds with an AuthorizeResponse with - idTokenInfo.status Accepted - idTokenInfo.groupIdToken.idToken <Configured groupIdToken>
	8. The Charging Station sends a TransactionEventRequest	9. The OCTT responds with a TransactionEventResponse with - idTokenInfo.status Accepted - idTokenInfo.groupIdToken.idToken <Configured groupIdToken>
	10. Execute Reusable State <i>EVConnectedPostSession</i>	
	11. Execute Reusable State <i>EVDDisconnected</i>	
	12. Execute Reusable State <i>ParkingBayUnoccupied</i>	

Test case name	Authorization by GroupId - Success
Tool validations	<ul style="list-style-type: none"> * Step 1: Message AuthorizeRequest - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type> * Step 3: Message TransactionEventRequest - triggerReason <i>Authorized</i> - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type> if transaction was already started - eventType <i>Updated</i> else - eventType <i>Started</i> * Step 6: Message AuthorizeRequest - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type> * Step 8: Message TransactionEventRequest - triggerReason <i>StopAuthorized</i> - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type>
	Post scenario validations: <ul style="list-style-type: none"> - N/a

Table 79. Test Case Id: TC_C_41_CS

Test case name	Authorization by GroupId - Success with Authorization Cache	
Test case Id	TC_C_41_CS	
Use case Id(s)	C09	
Requirement(s)	C09_FR_02, C09_FR_03, C09_FR_07	
System under test	Charging Station	
Description	This test case covers how a Charging Station can authorize an action for an EV Driver based on GroupId information. This could for example be used if 2 people regularly use the same EV: they can use their own IdToken (e.g. RFID card), and can deauthorize transactions that were started with the other idToken (with the same GroupId).	
Purpose	To verify if the Charging Station is able to correctly handle the Authorization of idTokens with the same GroupId when stored in the Authorization Cache according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	- AuthCacheCtrlr.AuthCacheAvailable is implemented with value true	
Before (Preparations)	<p>Configuration State: AuthCacheEnabled is true (if implemented) LocalPreAuthorize is true (if implemented)</p> <p>Memory State: <i>IdTokenCached</i> for <Configured valid IdToken fields> <i>IdTokenCached</i> for <Configured valid IdToken2 fields></p> <p>Reusable State(s): State is <i>EVConnectedPreSession</i></p>	
Main (Test scenario)	<p>Charging Station</p> <p><u>Manual Action:</u> Present valid idToken with <Configured GroupId> which is configured in the Authorization Cache</p> <p>1. The Charging Station sends a TransactionEventRequest <u>Note(s):</u> - This step needs to be executed when TxStartPoint contains Authorized OR the transaction already started. So in the case TxStartPoint contains ParkingBayOccupancy</p> <p>3. Execute Reusable State <i>EnergyTransferStarted</i> <u>Manual Action:</u> Present valid idToken2 with <Configured GroupId> which is configured in the Authorization Cache</p> <p>4. Execute Reusable State <i>StopAuthorized</i> 5. Execute Reusable State <i>EVConnectedPostSession</i> 6. Execute Reusable State <i>EVDDisconnected</i> 7. Execute Reusable State <i>ParkingBayUnoccupied</i></p>	<p>CSMS</p> <p>2. The OCTT responds with a TransactionEventResponse with - idTokenInfo.status Accepted - idTokenInfo.groupIdToken.idToken <Configured groupIdToken></p>
Tool validations	<ul style="list-style-type: none"> * Step 1: Message TransactionEventRequest <ul style="list-style-type: none"> - triggerReason Authorized - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type> if transaction was already started <ul style="list-style-type: none"> - eventType Updated else <ul style="list-style-type: none"> - eventType Started * Step 4: Message TransactionEventRequest <ul style="list-style-type: none"> - triggerReason StopAuthorized - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type> <p>Post scenario validations: - N/a</p>	

Table 80. Test Case Id: TC_C_42_CS

Test case name	Authorization by GroupId - Not stopped by GroupId	
Test case Id	TC_C_42_CS	
Use case Id(s)	C09	
Requirement(s)	C09_FR_11	
System under test	Charging Station	
Description	This test case covers how a Charging Station can authorize an action for an EV Driver based on GroupId information. This could for example be used if 2 people regularly use the same EV: they can use their own IdToken (e.g. RFID card), and can deauthorize transactions that were started with the other idToken (with the same GroupId).	
Purpose	To verify if the Charging Station is able to correctly handle the Authorization of idTokens with the same GroupId, while one of them is invalid, according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	- N/a	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): State is <i>EVConnectedPreSession</i>	
Main (Test scenario)	Charging Station <u>Manual Action:</u> Present valid idToken with <Configured GroupId>	CSMS
	1. The Charging Station sends an AuthorizeRequest 3. The Charging Station sends a TransactionEventRequest <u>Note(s):</u> - This step needs to be executed when TxStartPoint contains Authorized OR the transaction already started. So in the case TxStartPoint contains ParkingBayOccupancy	2. The OCTT responds with an AuthorizeResponse with - idTokenInfo.status Accepted - idTokenInfo.groupIdToken.idToken <Configured groupIdToken> 4. The OCTT responds with a TransactionEventResponse with - idTokenInfo.status Accepted - idTokenInfo.groupIdToken.idToken <Configured groupIdToken>
	5. Execute Reusable State <i>EnergyTransferStarted</i> <u>Manual Action:</u> Present invalid idToken with <Configured GroupId>	6. The Charging Station sends an AuthorizeRequest 7. The OCTT responds with an AuthorizeResponse with - idTokenInfo.status Invalid - idTokenInfo.groupIdToken.idToken <Configured groupIdToken>
	<u>Note(s):</u> OCTT will wait to see if CS indeed doesn't send a TransactionEventRequest	

Test case name	Authorization by GroupId - Not stopped by GroupId
Tool validations	<ul style="list-style-type: none"> * Step 1: Message AuthorizeRequest - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type> * Step 3: Message TransactionEventRequest - triggerReason Authorized - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type> if transaction was already started - eventType Updated else - eventType Started * Step 6: Message AuthorizeRequest - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type>
	<p>Post scenario validations:</p> <ul style="list-style-type: none"> - The energy transfer is not stopped

Table 81. Test Case Id: TC_C_44_CS

Test case name	Authorization by GroupId - Invalid status with Authorization Cache	
Test case Id	TC_C_44_CS	
Use case Id(s)	C09	
Requirement(s)	C09_FR_02, C09_FR_03, C09_FR_07	
System under test	Charging Station	
Description	This test case covers how a Charging Station can authorize an action for an EV Driver based on GroupId information. This could for example be used if 2 people regularly use the same EV: they can use their own IdToken (e.g. RFID card), and can deauthorize transactions that were started with the other idToken (with the same GroupId).	
Purpose	To verify if the Charging Station is able to correctly handle the Authorization of idTokens with the same GroupId when stored in the Authorization Cache, but one of them is invalid, according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	- AuthCacheCtrlr.AuthCacheAvailable is implemented with value true	
Before (Preparations)	<p>Configuration State: AuthCacheEnabled is true (If implemented) LocalPreAuthorize is true (If implemented) AuthCacheCtrlrDisablePostAuthorize is false (If implemented)</p> <p>Memory State: <i>IdTokenCached</i> for <Configured valid IdToken fields> <i>IdTokenCached</i> for <Configured invalid IdToken fields></p> <p>Reusable State(s): State is <i>EVConnectedPreSession</i></p>	
Main (Test scenario)	<p>Charging Station</p> <p>Manual Action: Present valid idToken with <Configured GroupId> which is configured in the Authorization Cache</p> <p>1. The Charging Station sends a TransactionEventRequest Note(s): - This step needs to be executed when TxStartPoint contains Authorized OR the transaction already started. So in the case TxStartPoint contains ParkingBayOccupancy</p> <p>3. Execute Reusable State <i>EnergyTransferStarted</i> Manual Action: Present invalid idToken with <Configured GroupId> which is configured in the Authorization Cache</p> <p>4. The Charging Station sends an AuthorizeRequest</p>	<p>CSMS</p> <p>2. The OCTT responds with a TransactionEventResponse with - idTokenInfo.status Accepted - idTokenInfo.groupIdToken.idToken <Configured groupIdToken></p> <p>5. The OCTT responds with an AuthorizeResponse with - idTokenInfo.status Invalid - idTokenInfo.groupIdToken.idToken <Configured groupIdToken></p> <p>Note(s): OCTT will wait to see if CS indeed doesn't send a TransactionEventRequest</p>

Test case name	Authorization by GroupId - Invalid status with Authorization Cache
Tool validations	<ul style="list-style-type: none"> * Step 1: Message TransactionEventRequest <ul style="list-style-type: none"> - triggerReason <i>Authorized</i> - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type> if transaction was already started <ul style="list-style-type: none"> - eventType <i>Updated</i> else - eventType <i>Started</i> * Step 4: Message AuthorizeRequest <ul style="list-style-type: none"> - idToken.idToken <Configured invalid_idtoken_idtoken> - idToken.type <Configured invalid_idtoken_type>
	Post scenario validations: <ul style="list-style-type: none"> - N/a

Table 82. Test Case Id: TC_C_45_CS

Test case name	Authorization by GroupId - Master pass - Not able to start transaction + groupId	
Test case Id	TC_C_45_CS	
Use case Id(s)	C09	
Requirement(s)	C16.FR.03	
System under test	Charging Station	
Description	This test case covers how a Charging Station can authorize an action for an EV Driver based on GroupId information. This could for example be used if 2 people regularly use the same EV: they can use their own IdToken (e.g. RFID card), and can deauthorize transactions that were started with the other idToken (with the same GroupId).	
Purpose	To verify if the Charging Station is able to correctly handle the Authorization of an idToken with the same GroupId as the MasterPassGroupId according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	- The Charging station supports MasterPass feature.	
Before (Preparations)	<p>Configuration State: TxCtrlr.TxStartPoint should contain <i>Authorized</i> or <i>PowerPathClosed</i> and not contain <i>ParkingBayOccupancy</i> or <i>EVConnected</i> AuthCtrlr.MasterPassGroupId is <Configured MasterPassGroupId></p> <p>Memory State: N/a</p> <p>Reusable State(s): N/a</p>	
Main (Test scenario)	<p>Charging Station</p> <p><u>Manual Action:</u> Present configured masterpass idToken</p> <p>1. The Charging Station sends an AuthorizeRequest</p>	<p>CSMS</p> <p>2. The OCTT responds with an AuthorizeResponse with</p> <ul style="list-style-type: none"> - idTokenInfo.status Accepted - idTokenInfo.groupIdToken.idToken <Configured masterPassGroupId> <p><u>Note:</u> The Charging Station will not authorize the transaction and send a TransactionEventRequest (in case of TxStartPoint Authorized).</p> <p>3. Execute Reusable State <i>EVConnectedPreSession</i></p> <p>4. The Charging Station will NOT send a TransactionEventRequest with chargingState <i>Charging</i> and triggerReason <i>ChargingStateChanged</i></p>
Tool validations	<ul style="list-style-type: none"> * Step 1: Message AuthorizeRequest - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type> <p>Post scenario validations: - N/a</p>	

Table 83. Test Case Id: TC_C_46_CS

Test case name	Store Authorization Data in the Authorization Cache - AuthCacheLifeTime	
Test case Id	TC_C_46_CS	
Use case Id(s)	C10	
Requirement(s)	C10_FR_08	
System under test	Charging Station	
Description	This test case covers how the Charging Station autonomously stores a record of previously presented identifiers that have been successfully authorized by the CSMS in the Authorization Cache. (Successfully meaning: a response received on a message containing an IdToken)	
Purpose	To verify if the Charging Station is able to correctly remove an idToken when this one is not reused again within the specified amount of time (AuthCacheLifeTime) according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	<ul style="list-style-type: none"> - AuthCacheCtrlr.AuthCacheAvailable is implemented with value true - Configuration variable AuthCacheLifeTime is implemented 	
Before (Preparations)	<p>Configuration State: AuthCacheLifeTime is <i><Configured TransactionDuration></i> AuthCacheCtrlr.LocalPreAuthorize is true (If implemented)</p> <p>Memory State: <i>IdTokenCached</i> <Configured valid idtoken fields></p> <p>Reusable State(s): State is <i>EVConnectedPreSession</i></p>	
Main (Test scenario)	Charging Station	CSMS
	<ol style="list-style-type: none"> 1. Wait for <i><Configured Transaction Duration></i> seconds 2. Execute Reusable State <i>Authorized</i> (local) 	
Tool validations	<p>N/a</p> <p>Post scenario validations: - N/a</p>	

Table 84. Test Case Id: TC_C_47_CS

Test case name	Stop Transaction with a Master Pass - With UI - All transactions																	
Test case Id	TC_C_47_CS																	
Use case Id(s)	C16																	
Requirement(s)	C16_FR_01																	
System under test	Charging Station																	
Description	This test case covers how somebody with a Master Pass (User) can stop (selected) ongoing transactions, so the cable becomes unlocked. This Master Pass can be configured in: MasterPassGroupId. This could for example be usefull for Law Enforcement officials.																	
Purpose	To verify if the Charging Station is able to correctly stop all transactions when an idToken which has the MasterPass as GroupId is used and the user has selected to stop all transactions in the User Interface according to the mechanism as described in the OCPP specification.																	
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station supports at least one authorization method described at the following Use cases; C01, C04. - Charging station has a User Interface 																	
Before (Preparations)	<p>Configuration State: AuthCtrlr.MastersPassGroupId is configured</p> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>EnergyTransferStarted</i> for all EVSE</p>																	
Main (Test scenario)	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="padding: 5px;">Charging Station</td> <td style="padding: 5px;">CSMS</td> </tr> <tr> <td colspan="2" style="padding: 5px;"><u>Manual Action:</u> Present configured masterpass idToken</td> </tr> <tr> <td style="padding: 5px;">1. The Charging Station sends an AuthorizeRequest</td> <td style="padding: 5px;">2. The OCTT responds with an AuthorizeResponse with idTokenInfo.status Accepted idTokenInfo.groupIdToken.idToken <Configured masterPassGroupId></td> </tr> <tr> <td colspan="2" style="padding: 5px;"><u>Manual Action:</u> Select to stop all transactions</td> </tr> <tr> <td style="padding: 5px;">3. The Charging Station sends a TransactionEventRequest for all EVSE</td> <td style="padding: 5px;">4. The OCTT responds with a TransactionEventResponse with idTokenInfo.status Accepted idTokenInfo.groupIdToken.idToken <Configured masterPassGroupId> for all EVSE</td> </tr> <tr> <td colspan="2" style="padding: 5px;">5. Execute Reusable State <i>EVConnectedPostSession</i> for all EVSE</td> </tr> <tr> <td colspan="2" style="padding: 5px;">6. Execute Reusable State <i>EVDDisconnected</i> for all EVSE</td> </tr> <tr> <td colspan="2" style="padding: 5px;">7. Execute Reusable State <i>ParkingBayUnoccupied</i> for all EVSE</td> </tr> </table>	Charging Station	CSMS	<u>Manual Action:</u> Present configured masterpass idToken		1. The Charging Station sends an AuthorizeRequest	2. The OCTT responds with an AuthorizeResponse with idTokenInfo.status Accepted idTokenInfo.groupIdToken.idToken <Configured masterPassGroupId>	<u>Manual Action:</u> Select to stop all transactions		3. The Charging Station sends a TransactionEventRequest for all EVSE	4. The OCTT responds with a TransactionEventResponse with idTokenInfo.status Accepted idTokenInfo.groupIdToken.idToken <Configured masterPassGroupId> for all EVSE	5. Execute Reusable State <i>EVConnectedPostSession</i> for all EVSE		6. Execute Reusable State <i>EVDDisconnected</i> for all EVSE		7. Execute Reusable State <i>ParkingBayUnoccupied</i> for all EVSE		
Charging Station	CSMS																	
<u>Manual Action:</u> Present configured masterpass idToken																		
1. The Charging Station sends an AuthorizeRequest	2. The OCTT responds with an AuthorizeResponse with idTokenInfo.status Accepted idTokenInfo.groupIdToken.idToken <Configured masterPassGroupId>																	
<u>Manual Action:</u> Select to stop all transactions																		
3. The Charging Station sends a TransactionEventRequest for all EVSE	4. The OCTT responds with a TransactionEventResponse with idTokenInfo.status Accepted idTokenInfo.groupIdToken.idToken <Configured masterPassGroupId> for all EVSE																	
5. Execute Reusable State <i>EVConnectedPostSession</i> for all EVSE																		
6. Execute Reusable State <i>EVDDisconnected</i> for all EVSE																		
7. Execute Reusable State <i>ParkingBayUnoccupied</i> for all EVSE																		
Tool validations	<ul style="list-style-type: none"> * Step 1: Message AuthorizeRequest - idToken.idToken <Configured masterpass_idtoken_idtoken> - idToken.type <Configured masterpass_idtoken_type> * Step 3: Message TransactionEventRequest - transactionInfo.stoppedReason MasterPass - idToken omit or - idToken.idToken <Configured masterpass_idtoken_idtoken> and - idToken.type <Configured masterpass_idtoken_type> <p>Post scenario validations: - N/a</p>																	

Table 85. Test Case Id: TC_C_48_CS

Test case name	Stop Transaction with a Master Pass - With UI - With UI - Specific transactions																
Test case Id	TC_C_48_CS																
Use case Id(s)	C16																
Requirement(s)	C16_FR_01																
System under test	Charging Station																
Description	This test case covers how somebody with a Master Pass (User) can stop (selected) ongoing transactions, so the cable becomes unlocked. This Master Pass can be configured in: MasterPassGroupId. This could for example be usefull for Law Enforcement officials.																
Purpose	To verify if the Charging Station is able to correctly stop a transaction when an idToken which has the MasterPass as GroupId is used and the user has selected to stop one transaction in the User Interface according to the mechanism as described in the OCPP specification.																
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station supports at least one authorization method described at the following Use cases; C01, C04. - Charging station has a User Interface 																
Before (Preparations)	<p>Configuration State: AuthCtrlr.MastersPassGroupId is configured</p> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>EnergyTransferStarted</i> for all EVSE</p>																
Main (Test scenario)	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="padding: 5px;">Charging Station</td> <td style="padding: 5px;">CSMS</td> </tr> <tr> <td colspan="2" style="padding: 5px;"><u>Manual Action:</u> Present configured masterpass idToken</td> </tr> <tr> <td style="padding: 5px;">1. The Charging Station sends an AuthorizeRequest</td> <td style="padding: 5px;">2. The OCTT responds with an AuthorizeResponse with idTokenInfo.status Accepted idTokenInfo.groupIdToken.idToken <Configured masterPassGroupId></td> </tr> <tr> <td colspan="2" style="padding: 5px;"><u>Manual Action:</u> Select to stop the transaction on EVSE 1</td> </tr> <tr> <td style="padding: 5px;">3. The Charging Station sends a TransactionEventRequest</td> <td style="padding: 5px;">4. The OCTT responds with a TransactionEventResponse with idTokenInfo.status Accepted idTokenInfo.groupIdToken.idToken <Configured masterPassGroupId></td> </tr> <tr> <td colspan="2" style="padding: 5px;">5. Execute Reusable State <i>EVConnectedPostSession</i></td> </tr> <tr> <td colspan="2" style="padding: 5px;">6. Execute Reusable State <i>EVDisconnected</i></td> </tr> <tr> <td colspan="2" style="padding: 5px;">7. Execute Reusable State <i>ParkingBayUnoccupied</i></td> </tr> </table>	Charging Station	CSMS	<u>Manual Action:</u> Present configured masterpass idToken		1. The Charging Station sends an AuthorizeRequest	2. The OCTT responds with an AuthorizeResponse with idTokenInfo.status Accepted idTokenInfo.groupIdToken.idToken <Configured masterPassGroupId>	<u>Manual Action:</u> Select to stop the transaction on EVSE 1		3. The Charging Station sends a TransactionEventRequest	4. The OCTT responds with a TransactionEventResponse with idTokenInfo.status Accepted idTokenInfo.groupIdToken.idToken <Configured masterPassGroupId>	5. Execute Reusable State <i>EVConnectedPostSession</i>		6. Execute Reusable State <i>EVDisconnected</i>		7. Execute Reusable State <i>ParkingBayUnoccupied</i>	
Charging Station	CSMS																
<u>Manual Action:</u> Present configured masterpass idToken																	
1. The Charging Station sends an AuthorizeRequest	2. The OCTT responds with an AuthorizeResponse with idTokenInfo.status Accepted idTokenInfo.groupIdToken.idToken <Configured masterPassGroupId>																
<u>Manual Action:</u> Select to stop the transaction on EVSE 1																	
3. The Charging Station sends a TransactionEventRequest	4. The OCTT responds with a TransactionEventResponse with idTokenInfo.status Accepted idTokenInfo.groupIdToken.idToken <Configured masterPassGroupId>																
5. Execute Reusable State <i>EVConnectedPostSession</i>																	
6. Execute Reusable State <i>EVDisconnected</i>																	
7. Execute Reusable State <i>ParkingBayUnoccupied</i>																	
Tool validations	<ul style="list-style-type: none"> * Step 1: Message AuthorizeRequest - idToken.idToken <Configured masterpass_idtoken_idtoken> - idToken.type <Configured masterpass_idtoken_type> * Step 3: Message TransactionEventRequest - transactionInfo.stoppedReason MasterPass - idToken omit or - idToken.idToken <Configured masterpass_idtoken_idtoken> and - idToken.type <Configured masterpass_idtoken_type> <p>Post scenario validations: - All other EVSE still transfer energy</p>																

Table 86. Test Case Id: TC_C_49_CS

Test case name	Stop Transaction with a Master Pass - Without UI	
Test case Id	TC_C_49_CS	
Use case Id(s)	C16	
Requirement(s)	C16_FR_02	
System under test	Charging Station	
Description	This test case covers how somebody with a Master Pass (User) can stop (selected) ongoing transactions, so the cable becomes unlocked. This Master Pass can be configured in: MasterPassGroupId. This could for example be usefull for Law Enforcement officials.	
Purpose	To verify if the Charging Station is able to correctly stop all transactions when an idToken which has the MasterPass as GroupId is used and the Charging station does not have an User Interface according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	- The Charging Station supports at least one authorization method described at the following Use cases; C01, C04.	
Before (Preparations)	<p>Configuration State: AuthCtrlr.MastersPassGroupId is configured</p> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>EnergyTransferStarted</i> for EVSEId 1 and EVSEId 2 if the Charging Station has more than one EVSE. With: - <Configured valid_idtoken> for EVSE 1 - <Configured valid_idtoken2> for EVSE 2</p>	
Main (Test scenario)	<p>Charging Station</p> <p><u>Manual Action:</u> Present configured masterpass idToken</p> <ol style="list-style-type: none"> 1. The Charging Station sends an AuthorizeRequest 2. The OCTT responds with an AuthorizeResponse with idTokenInfo.status Accepted idTokenInfo.groupIdToken.idToken <Configured masterPassGroupId> 3. The Charging Station sends a TransactionEventRequest for EVSE 1 (and 2) 4. The OCTT responds with a TransactionEventResponse with idTokenInfo.status Accepted idTokenInfo.groupIdToken.idToken <Configured masterPassGroupId> for EVSE 1 (and 2) 5. Execute Reusable State <i>EVConnectedPostSession</i> for EVSE 1 (and 2) 6. Execute Reusable State <i>EVDDisconnected</i> for EVSE 1 (and 2) 7. Execute Reusable State <i>ParkingBayUnoccupied</i> for EVSE 1 (and 2) 	
Tool validations	<ul style="list-style-type: none"> * Step 1: Message AuthorizeRequest - idToken.idToken <Configured masterpass_idtoken_idtoken> - idToken.type <Configured masterpass_idtoken_type> * Step 3: Message TransactionEventRequest - transactionInfo.stoppedReason MasterPass - idToken omit or - idToken.idToken <Configured masterpass_idtoken_idtoken> and - idToken.type <Configured masterpass_idtoken_type> <p>Post scenario validations: - N/a</p>	

Table 87. Test Case Id: TC_C_26_CS

Test case name	Offline Authorization - Unknown Id	
Test case Id	TC_C_26_CS	
Use case Id(s)	C15 & C13	
Requirement(s)	C15.FR.02,C15.FR.06,C15.FR.08,C13.FR.04	
System under test	Charging Station	
Description	The Charging Station is allowed to allow starting a transaction for unknown idTokens when offline and configured to do so.	
Purpose	To verify if the Charging Station is able to start a transaction while being offline for an unknown idToken, when it is configured to do so.	
Prerequisite(s)	OfflineTxForUnknownIdEnabled is implemented.	
Before (Preparations)	<p>Configuration State:</p> <p>LocalAuthListEnabled is <i>true</i> (If implemented)</p> <p>LocalPreAuthorize is <i>true</i> (If implemented)</p> <p>OfflineTxForUnknownIdEnabled is <i>true</i></p> <p>LocalAuthorizeOffline is <i>true</i></p> <p>MaxEnergyOnInvalidId is <i>0</i> (If implemented)</p> <p>StopTxOnInvalidId is <i>false</i></p> <p>Memory State:</p> <p>N/a</p> <p>Reusable State(s): State is <i>StartOfflineTransaction</i></p>	
Main (Test scenario)	<p>Charging Station</p> <p>1. The Charging Stations sends a TransactionEventRequest</p> <p>Note(s): - <i>The Charging Station will empty its Transaction message queue. This will contain one or more TransactionEventRequest messages</i></p> <p>Manual Action: Present valid idToken.</p> <p>Manual Action: Unplug cable</p> <p>3. The Charging Stations sends a TransactionEventRequest with triggerReason StopAuthorized</p>	<p>CSMS</p> <p>2. The OCTT responds with a TransactionEventResponse</p> <p>Note(s): - <i>The OCTT will respond to the TransactionEventRequest containing the idToken, with idtokenInfo.status Invalid</i></p> <p>4. The OCTT responds with a TransactionEventResponse</p>
Tool validations	<ul style="list-style-type: none"> * Step 1: All Message(s): TransactionEventRequest - offline must be <i>true</i> * Step 1: One of the Message(s): TransactionEventRequest - chargingState must be <i>SuspendedEVSE</i> <p>Post scenario validations: N/a</p>	

Table 88. Test Case Id: TC_C_56_CS

Test case name	Local start transaction - Authorization Unknown	
Test case Id	TC_C_56_CS	
Use case Id(s)	C01	
Requirement(s)	C01.FR.02	
System under test	Charging Station	
Description	When a Charging Station needs to charge an EV, it needs to authorize the EV Driver first at the CSMS before the charging can be started or stopped.	
Purpose	To verify whether the Charging Station is able to handle receiving an Unknown idToken.	
Prerequisite(s)	The Charging Station does NOT have a cable lock, which prevents the EV driver to connect the EV and EVSE before authorization.	
Before (Preparations)	<p>Configuration State: AuthCtrlIr.AuthEnabled is <i>true</i> (If implemented AND ReadWrite) AuthCtrlIr.DisableRemoteAuthorization is <i>false</i> (If implemented)</p> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>EVConnectedPreSession</i></p>	
Main (Test scenario)	<p>Charging Station</p> <p><u>Manual Action:</u> Present invalid idToken.</p> <p>1. The Charging Station sends an AuthorizeRequest</p>	<p>CSMS</p> <p>2. The OCTT responds with an AuthorizeResponse with idTokenInfo.status Unknown</p>
	<p>Note(s):</p> <ul style="list-style-type: none"> - The Charging Station SHALL NOT send a TransactionEventRequest message after the AuthorizeRequest from step 1. - The OCTT waits <Configured message timeout> seconds, before ending the testcase. 	
Tool validations	<ul style="list-style-type: none"> * Step 1: Message: AuthorizeRequest - idToken.idToken <Configured invalid_idtoken_idtoken> - idToken.type <Configured invalid_idtoken_type> <p>Post scenario validations: N/a</p>	

2.5. D Local Authorization List Management

This section is intentionally blank, this will be added in a later version.

2.6. E Transactions

Table 89. Test Case Id: TC_E_01_CS

Test case name	Start transaction options - PowerPathClosed	
Test case Id	TC_E_01_CS	
Use case Id(s)	E01(S5)	
Requirement(s)	E01.FR.05, E01.FR.07, E01.FR.10, E01.FR.15, E01.FR.16	
System under test	Charging Station	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the start options.	
Purpose	To verify if the Charging Station starts a transaction when the power path has been closed and it has been configured to do so.	
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station does NOT have the following configuration; The mutability of TxStartPoint is <i>ReadOnly</i> AND (the value <i>PowerPathClosed</i> is NOT set OR (<i>ParkingBayOccupancy</i> OR <i>EVConnected</i> OR <i>Authorized</i> OR <i>DataSigned</i>), is set). - If the mutability of TxStartPoint is <i>ReadWrite</i>, then the value <i>PowerPathClosed</i> must be supported. 	
Before (Preparations)	Configuration State: If the mutability of TxStartPoint is <i>ReadWrite</i> then TxStartPoint contains <i>PowerPathClosed</i> Memory State: N/a	
	Reusable State(s): State is <i>Authorized</i>	
Main (Test scenario)	Charging Station	CSMS
	1. Execute Reusable State <i>EVConnectedPreSession</i>	
Tool validations	N/a Post scenario validations: N/a	

Table 90. Test Case Id: TC_E_02_CS

Test case name	Start transaction options - EnergyTransfer	
Test case Id	TC_E_02_CS	
Use case Id(s)	E01(S6)	
Requirement(s)	E01.FR.06, E01.FR.07, E01.FR.10, E01.FR.15, E01.FR.16	
System under test	Charging Station	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the start options.	
Purpose	To verify if the Charging Station starts a transaction when the energy transfer starts and it has been configured to do so.	
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station does NOT have the following configuration; The mutability of TxStartPoint is ReadOnly AND (the value EnergyTransfer is NOT set OR (ParkingBayOccupancy OR EVConnected OR Authorized OR DataSigned OR PowerPathClosed), is set). - If the mutability of TxStartPoint is ReadWrite, then the value EnergyTransfer must be supported. 	
Before (Preparations)	<p>Configuration State: If the mutability of TxStartPoint is ReadWrite then TxStartPoint contains EnergyTransfer</p> <p>Memory State: N/a</p> <p>Reusable State(s): State is Authorized</p>	
Main (Test scenario)	<p>Charging Station</p> <p>Manual Action: Connect the EV and EVSE.</p> <p>1. The Charging Station notifies the CSMS about the status change of the connector.</p> <p>3. The Charging Station sends a TransactionEventRequest</p>	<p>CSMS</p> <p>2. The OCTT responds accordingly.</p> <p>4. The OCTT responds with a TransactionEventResponse</p>
Tool validations	<p>* Step 1: Message: StatusNotificationRequest - connectorStatus must be <i>Occupied</i></p> <p>Message: NotifyEventRequest - eventData[0].trigger must be <i>Delta</i> - eventData[0].actualValue must be <i>Occupied</i> - eventData[0].component.name must be <i>Connector</i> - eventData[0].variable.name must be <i>AvailabilityState</i></p> <p>* Step 3: Message: TransactionEventRequest - eventType must be <i>Started</i> - If the OCTT is configured to start transactions using a RequestStartTransactionRequest message then triggerReason must be <i>RemoteStart</i> Else triggerReason must be <i>ChargingStateChanged</i> or <i>Authorized</i> - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type> - evse must be provided - evse.connectorId must be provided - transactionInfo.chargingState must be <i>Charging</i></p> <p>Post scenario validations: N/a</p>	

Table 91. Test Case Id: TC_E_09_CS

Test case name	Start transaction options - EVConnected									
Test case Id	TC_E_09_CS									
Use case Id(s)	E01(S2)									
Requirement(s)	E01.FR.02, E01.FR.07, E01.FR.10, E01.FR.15, E01.FR.16									
System under test	Charging Station									
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the start options.									
Purpose	To verify if the Charging Station starts a transaction when the EV and EVSE are connected and it has been configured to do so.									
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station does NOT have the following configuration; The mutability of TxStartPoint is <i>ReadOnly</i> AND (the value <i>EVConnected</i> is NOT set OR <i>ParkingBayOccupancy</i> is set). - If the mutability of TxStartPoint is <i>ReadWrite</i>, then the value <i>EVConnected</i> must be supported. 									
Before (Preparations)	<p>Configuration State: If the mutability of TxStartPoint is <i>ReadWrite</i> then TxStartPoint contains <i>EVConnected</i></p> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>ParkingBayOccupied</i></p>									
Main (Test scenario)	<table border="1"> <tr> <td>Charging Station</td> <td>CSMS</td> </tr> <tr> <td><u>Manual Action:</u> Connect the EV and EVSE.</td> <td></td> </tr> <tr> <td>1. The Charging Station notifies the CSMS about the status change of the connector.</td> <td>2. The OCTT responds accordingly.</td> </tr> <tr> <td>3. The Charging Station sends a TransactionEventRequest</td> <td>4. The OCTT responds with a TransactionEventResponse</td> </tr> </table>	Charging Station	CSMS	<u>Manual Action:</u> Connect the EV and EVSE.		1. The Charging Station notifies the CSMS about the status change of the connector.	2. The OCTT responds accordingly.	3. The Charging Station sends a TransactionEventRequest	4. The OCTT responds with a TransactionEventResponse	
Charging Station	CSMS									
<u>Manual Action:</u> Connect the EV and EVSE.										
1. The Charging Station notifies the CSMS about the status change of the connector.	2. The OCTT responds accordingly.									
3. The Charging Station sends a TransactionEventRequest	4. The OCTT responds with a TransactionEventResponse									
Tool validations	<ul style="list-style-type: none"> * Step 1: Message: StatusNotificationRequest - connectorStatus must be <i>Occupied</i> Message: NotifyEventRequest - eventType[0].trigger must be <i>Delta</i> - eventType[0].actualValue must be <i>Occupied</i> - eventType[0].component.name must be <i>Connector</i> - eventType[0].variable.name must be <i>AvailabilityState</i> * Step 3: Message: TransactionEventRequest - eventType must be <i>Started</i> - triggerReason must be <i>CablePluggedIn</i> or <i>ChargingStateChanged</i> - evse must be provided - evse.connectorId must be provided - transactionInfo.chargingState must be <i>EVConnected</i> <p>Post scenario validations: N/a</p>									

Table 92. Test Case Id: TC_E_10_CS

Test case name	Start transaction options - Authorized - Local	
Test case Id	TC_E_10_CS	
Use case Id(s)	E01(S3) AND (C01 OR C02 OR C04 OR C06)	
Requirement(s)	E01.FR.03, E01.FR.07, E01.FR.10, E01.FR.15, E01.FR.16 AND C01.FR.02, C02.FR.01, C06.FR.02	
System under test	Charging Station	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the start options.	
Purpose	To verify if the Charging Station starts a transaction when the EV and EVSE are connected and it has been configured to do so.	
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station supports at least one authorization method described at the following Use cases; C01, C02, C04, C06. - The Charging Station does NOT have the following configuration; The mutability of TxStartPoint is <i>ReadOnly</i> AND the value Authorized is NOT set. - If the mutability of TxStartPoint is <i>ReadWrite</i>, then the value Authorized must be supported. 	
Before (Preparations)	<p>Configuration State: If the mutability of TxStartPoint is <i>ReadWrite</i> then TxStartPoint contains Authorized AuthCtrlr.AuthEnabled is <i>true</i> (If implemented AND <i>ReadWrite</i>) AuthCtrlr.DisableRemoteAuthorization is <i>false</i> (If implemented)</p> <p>Memory State: N/a</p> <p>Reusable State(s): N/a</p>	
Main (Test scenario)	Charging Station <u>Manual Action:</u> Present <i>IdToken</i> .	CSMS
	1. The Charging Station sends an AuthorizeRequest <u>Note(s):</u> - This step needs to be executed, unless AuthEnabled is implemented with mutability <i>ReadOnly</i> AND the value is set to <i>false</i> OR a start button as described at Use case C02 is used (This must be configured at the OCTT).	2. The OCTT responds with a AuthorizeResponse with idTokenInfo.status Accepted
	3. The Charging Station sends a TransactionEventRequest	4. The OCTT responds with a TransactionEventResponse
Tool validations	* Step 1: Message: AuthorizeRequest - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type> * Step 3: Message: TransactionEventRequest - eventType must be <i>Started</i> - triggerReason must be Authorized - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type>	
	Post scenario validations: N/a	

Table 93. Test Case Id: TC_E_13_CS

Test case name	Start transaction options - Authorized - Remote	
Test case Id	TC_E_13_CS	
Use case Id(s)	E01(S3) AND F02	
Requirement(s)	E01.FR.03 AND F01.FR.03, F01.FR.04, F01.FR.06, F01.FR.19, F02.FR.01	
System under test	Charging Station	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the start options.	
Purpose	To verify if the Charging Station starts a transaction when the EV and EVSE are connected and it has been configured to do so.	
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station does NOT have the following configuration; The mutability of TxStartPoint is ReadOnly AND the value Authorized is NOT set. - If the mutability of TxStartPoint is ReadWrite, then the value Authorized must be supported. 	
Before (Preparations)	<p>Configuration State: If the mutability of TxStartPoint is ReadWrite then TxStartPoint contains Authorized AuthCtrlr.AuthEnabled is true (If implemented AND ReadWrite) AuthCtrlr.DisableRemoteAuthorization is false (If implemented)</p> <p>Memory State: N/a</p> <p>Reusable State(s): N/a</p>	
Main (Test scenario)	<p>Charging Station</p> <p>2. The Charging Station responds with a RequestStartTransactionResponse</p> <p>Note(s): - This step needs to be executed when AuthCtrlr.AuthorizeRemoteStart is true, unless AuthEnabled is implemented with mutability ReadOnly AND the value is set to false.</p> <p>3. The Charging Station sends an AuthorizeRequest</p> <p>5. The Charging Station sends a TransactionEventRequest</p>	<p>CSMS</p> <p>1. The OCTT sends a RequestStartTransactionRequest with idToken.idToken <Configured valid_idtoken_idtoken> idToken.type <Configured valid_idtoken_type> evseld <Configured evseld></p> <p>4. The OCTT responds with an AuthorizeResponse with: idTokenInfo.status Accepted</p> <p>6. The OCTT responds with a TransactionEventResponse</p>
Tool validations	<ul style="list-style-type: none"> * Step 2: Message: RequestStartTransactionResponse - status must be Accepted * Step 3: Message: AuthorizeRequest - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type> * Step 5: Message: TransactionEventRequest - eventType must be Started - triggerReason must be RemoteStart - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type> - transactionInfo.remoteStartId must be present <p>Post scenario validations: N/a</p>	

Table 94. Test Case Id: TC_E_11_CS

Test case name	Start transaction options - DataSigned	
Test case Id	TC_E_11_CS	
Use case Id(s)	E01(S4)	
Requirement(s)	E01.FR.04, E01.FR.07, E01.FR.10, E01.FR.15, E01.FR.16	
System under test	Charging Station	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the start options.	
Purpose	To verify if the Charging Station starts a transaction when the EV and EVSE are connected and it has been configured to do so.	
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station does NOT have the following configuration; The mutability of TxStartPoint is ReadOnly AND (the value DataSigned is NOT set OR (ParkingBayOccupancy OR EVConnected OR Authorized), is set). - If the mutability of TxStartPoint is ReadWrite, then the value DataSigned must be supported. 	
Before (Preparations)	<p>Configuration State: If the mutability of TxStartPoint is ReadWrite then TxStartPoint contains DataSigned SampledDataCtrlr.SignReadings is true</p> <p>Memory State: N/a</p> <p>Reusable State(s): State is Authorized</p>	
Main (Test scenario)	<p>Charging Station</p> <p><u>Manual Action:</u> Connect the EV and EVSE.</p> <p>1. The Charging Station notifies the CSMS about the status change of the connector.</p> <p>3. The Charging Station sends a TransactionEventRequest</p> <p>5. The Charging Station sends a TransactionEventRequest</p>	<p>CSMS</p> <p>2. The OCTT responds accordingly.</p> <p>4. The OCTT responds with a TransactionEventResponse</p> <p>6. The OCTT responds with a TransactionEventResponse</p>

Test case name	Start transaction options - DataSigned
Tool validations	<ul style="list-style-type: none"> * Step 1: <ul style="list-style-type: none"> Message: StatusNotificationRequest - connectorStatus must be <i>Occupied</i> Message: NotifyEventRequest - eventData[0].trigger must be <i>Delta</i> - eventData[0].actualValue must be <i>Occupied</i> - eventData[0].component.name must be <i>Connector</i> - eventData[0].variable.name must be <i>AvailabilityState</i> <ul style="list-style-type: none"> * Step 3: <ul style="list-style-type: none"> Message: TransactionEventRequest - eventType must be <i>Started</i> - If the OCTT is configured to start transactions using a RequestStartTransactionRequest message then triggerReason must be <i>RemoteStart</i> or <i>SignedDataReceived</i> Else triggerReason must be <i>SignedDataReceived</i> - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type> - evse must be provided - evse.connectorId must be provided - meterValue is provided with the following values: <ul style="list-style-type: none"> sampledValue.context is <i>Transaction.Begin</i> sampledValue.signedMeterValue.encodingMethod is not omitted sampledValue.signedMeterValue.publicKey is not omitted sampledValue.signedMeterValue.signedMeterData is not omitted sampledValue.signedMeterValue.signingMethod is not omitted * Step 5: <ul style="list-style-type: none"> Message: TransactionEventRequest - eventType must be <i>Updated</i> - triggerReason must be <i>ChargingStateChanged</i> - transactionInfo.chargingState must be <i>Charging</i>
	<p>Post scenario validations: N/a</p>

Table 95. Test Case Id: TC_E_12_CS

Test case name	Start transaction options - ParkingBayOccupied	
Test case Id	TC_E_12_CS	
Use case Id(s)	E01(S1)	
Requirement(s)	E01.FR.01, E01.FR.07, E01.FR.10, E01.FR.15, E01.FR.16	
System under test	Charging Station	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the start options.	
Purpose	To verify if the Charging Station starts a transaction when the EV and EVSE are connected and it has been configured to do so.	
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station does NOT have the following configuration; The mutability of TxStartPoint is <i>ReadOnly</i> AND the value <i>ParkingBayOccupancy</i> is NOT set. - If the mutability of TxStartPoint is <i>ReadWrite</i>, then the value <i>ParkingBayOccupancy</i> must be supported. 	
Before (Preparations)	<p>Configuration State: If the mutability of TxStartPoint is <i>ReadWrite</i> then TxStartPoint contains <i>ParkingBayOccupancy</i></p> <p>Memory State: N/a</p> <p>Reusable State(s): N/a</p>	
Main (Scenario)	<p>Charging Station</p> <p>Manual Action: Drive EV into parking bay.</p>	<p>CSMS</p> <p>1. The Charging Station sends a TransactionEventRequest</p> <p>2. The OCTT responds with a TransactionEventResponse</p>
Tool validations	<p>* Step 1: Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - eventType must be <i>Started</i> - triggerReason must be <i>EVDetected</i> <p>Post scenario validations: N/a</p>	

Table 96. Test Case Id: TC_E_16_CS

Test case name	Stop transaction options - Deauthorized - Invalid idToken	
Test case Id	TC_E_16_CS	
Use case Id(s)	E06(S3)	
Requirement(s)	E06.FR.04, E06.FR.15 & C15.FR.04	
System under test	Charging Station	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the start options.	
Purpose	To verify if the Charging Station stops a transaction when the transaction gets deauthorized by the status from the idTokenInfo at a TransactionEventResponse message and it has been configured to do so.	
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station does NOT have the following configuration; The mutability of TxStopPoint is <i>ReadOnly</i> AND the value <i>Authorized</i> OR <i>PowerPathClosed</i> is NOT set OR (<i>EnergyTransfer</i> OR <i>DataSigned</i> is set). - If the mutability of TxStopPoint is <i>ReadWrite</i>, then the value <i>Authorized</i> OR <i>PowerPathClosed</i> must be supported. - The Charging Station supports local start/stop transaction. 	
Before (Preparations)	<p>Configuration State: If the mutability of TxStopPoint is <i>ReadWrite</i> then TxStopPoint contains <i>PowerPathClosed</i> AND/OR <i>Authorized</i> AuthCtrlr.AuthEnabled is <i>true</i> (If implemented AND <i>ReadWrite</i>) AuthCtrlr.DisableRemoteAuthorization is <i>false</i> (If implemented) OfflineTxForUnknownIdEnabled is <i>true</i> (If implemented) StopTxOnInvalidId is <i>true</i></p> <p>Memory State: <i>IdTokenCached</i> for <Configured valid idtoken fields> (If implemented) <i>IdTokenLocalAuthList</i> for <Configured valid idtoken fields> (If implemented)</p> <p>Reusable State(s): State is <i>StartOfflineTransaction</i></p>	
Main (Test scenario)	<p>Charging Station</p> <p>1. The Charging Stations sends a TransactionEventRequest</p> <p><u>Note(s):</u> - The Charging Station will empty its Transaction message queue. This will contain one or more TransactionEventRequest messages</p> <p>3. The Charging Stations sends a TransactionEventRequest</p> <p><u>Note(s):</u> - After having emptied its queue, the Charging Station will send a TransactionEventRequest in which it reports it deauthorizes the transaction.</p>	<p>CSMS</p> <p>2. The OCTT responds with a TransactionEventResponse</p> <p><u>Note(s):</u> - The OCTT will respond to the TransactionEventRequest containing the idToken, with idtokenInfo.status Invalid</p> <p>4. The OCTT responds with a TransactionEventResponse</p>
Tool validations	<ul style="list-style-type: none"> * Step 1: Message: TransactionEventRequest - offline must be <i>true</i> * Step 3: Message: TransactionEventRequest - eventType must be <i>Ended</i> - triggerReason must be <i>Deauthorized</i> - transactionInfo.stoppedReason is <i>DeAuthorized</i> <p>Post scenario validations: N/a</p>	

Table 97. Test Case Id: TC_E_17_CS

Test case name	Stop transaction options - Deauthorized - EV side disconnect	
Test case Id	TC_E_17_CS	
Use case Id(s)	E06(S3)	
Requirement(s)	E06.FR.04, E06.FR.15	
System under test	Charging Station	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the start options.	
Purpose	To verify if the Charging Station stops a transaction when the transaction gets deauthorized by a connection loss from the EV side and it has been configured to do so.	
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station does NOT have the following configuration; The mutability of TxStopPoint is <i>ReadOnly</i> AND the value <i>Authorized</i> OR <i>PowerPathClosed</i> is NOT set OR (<i>EnergyTransfer</i> OR <i>DataSigned</i> OR <i>EVConnected</i> is set). - If the mutability of TxStopPoint is <i>_ReadWrite</i>, then the value <i>Authorized</i> OR <i>PowerPathClosed</i> must be supported. 	
Before (Preparations)	<p>Configuration State: If the mutability of TxStopPoint is <i>ReadWrite</i> then TxStopPoint contains <i>PowerPathClosed</i> AND/OR <i>Authorized</i> StopTxOnEVSideDisconnect is <i>true</i> UnlockOnEVSideDisconnect is <i>false</i> AuthCtrlr.AuthEnabled is <i>true</i> (If implemented AND <i>ReadWrite</i>) AuthCtrlr.DisableRemoteAuthorization is <i>false</i> (If implemented)</p> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>EnergyTransferSuspended</i></p>	
Main (Scenario)	Charging Station <u>Manual Action:</u> Disconnect the EV and EVSE on EV side (EVSE loses connection with EV).	CSMS <u>1. The Charging Station sends a TransactionEventRequest</u> <u>2. The OCTT responds with a TransactionEventResponse</u>
	<u>Manual Action:</u> Present the <i>IdToken</i> that was used to start the transaction.	
	<u>Note(s):</u> <ul style="list-style-type: none"> - This manual action needs to be executed when the Charging Station has a detachable cable on the Charging Station side. 	
	<u>Manual Action:</u> Disconnect the EV and EVSE on Charging Station side.	
	<u>Note(s):</u> <ul style="list-style-type: none"> - This manual action needs to be executed when the Charging Station has a detachable cable on the Charging Station side. 	
	<u>3. The Charging Station notifies the CSMS about the current state of the connector.</u>	<u>4. The OCTT responds accordingly.</u>

Test case name	Stop transaction options - Deauthorized - EV side disconnect
Tool validations	<ul style="list-style-type: none"> * Step 1: <ul style="list-style-type: none"> Message: TransactionEventRequest <ul style="list-style-type: none"> - triggerReason must be <i>EVCommunicationLost</i> - transactionInfo.chargingState must be <i>Idle</i> - transactionInfo.stoppedReason must be <i>EVDisconnected</i> - eventType must be <i>Ended</i> * Step 3: <ul style="list-style-type: none"> Message: StatusNotificationRequest <ul style="list-style-type: none"> - connectorStatus <i>Available</i> Message: NotifyEventRequest <ul style="list-style-type: none"> - eventData[0].trigger <i>Delta</i> - eventData[0].actualValue "<i>Available</i>" - eventData[0].component.name "<i>Connector</i>" - eventData[0].variable.name "<i>AvailabilityState</i>"
	Post scenario validations: N/a

Table 98. Test Case Id: TC_E_39_CS

Test case name	Stop transaction options - Deauthorized - timeout	
Test case Id	TC_E_39_CS	
Use case Id(s)	E06(S3)	
Requirement(s)	E06.FR.04, E06.FR.15	
System under test	Charging Station	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the start options.	
Purpose	To verify if the Charging Station stops a transaction when the transaction gets deauthorized because the cable was not plugged in within the Configured durationout and it has been configured to do so.	
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station does NOT have the following configuration; The mutability of TxStopPoint is <i>ReadOnly</i> AND the value <i>Authorized</i> is NOT set. - If the mutability of TxStopPoint is <i>ReadWrite</i>, then the value <i>Authorized</i> must be supported. 	
Before (Preparations)	<p>Configuration State: If the mutability of TxStopPoint is <i>ReadWrite</i> then TxStopPoint contains <i>Authorized</i></p> <ul style="list-style-type: none"> - TxCtrlr.EVConnectionTimeOut is <i><Configured ev_connection_timeout></i> - AuthCtrlr.AuthEnabled is <i>true</i> (If implemented AND <i>ReadWrite</i>) AuthCtrlr.DisableRemoteAuthorization is <i>false</i> (If implemented) <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>Authorized</i> (local)</p>	
Main (Test scenario)	<p>Charging Station</p> <p>1. The Charging Station sends a TransactionEventRequest</p> <p>Note(s): - This step needs to be executed after the <i><Configured ev_connection_timeout></i> expires, if the transaction has been started. So in the case TxStartPoint contains <i>ParkingBayOccupancy OR Authorized</i></p>	<p>CSMS</p> <p>2. The OCTT responds with a TransactionEventResponse</p>
Tool validations	<p>* Step 1: Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - triggerReason must be <i>EVConnectTimeout</i> - eventType must be <i>Ended</i> - transactionInfo.stoppedReason must be <i>Timeout</i> <p>Post scenario validations: N/a</p>	

Table 99. Test Case Id: TC_E_03_CS

Test case name	Local start transaction - Cable plugin first - Success	
Test case Id	TC_E_03_CS	
Use case Id(s)	E02 AND (C01 OR C02 OR C04 OR C06)	
Requirement(s)	E02.FR.01, E02.FR.05, E02.FR.06, E02.FR.07, E02.FR.13, E02.FR.15, E02.FR.16, E02.FR.17, E01.FR.16 AND C01.FR.02, C02.FR.01, C06.FR.02	
System under test	Charging Station	
Description	OCPP 2.x.x allows an EV driver to either first connect the EV and EVSE OR present a form of identification. Both sequences will result in being able to charge.	
Purpose	To verify if the Charging Station is able to start a charging session when the EV driver first connects the EV and EVSE, before authorization.	
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station supports at least one authorization method described at the following Use cases; C01, C02, C04, C06. - The Charging Station does NOT have a cable lock, which prevents the EV driver to connect the EV and EVSE before authorization. 	
Before (Preparations)	<p>Configuration State: AuthCtrlr.AuthEnabled is <i>true</i> (If implemented AND ReadWrite) AuthCtrlr.DisableRemoteAuthorization is <i>false</i> (If implemented)</p> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>EVConnectedPreSession</i></p>	
Main (Test scenario)	<p>Charging Station</p> <p>1. Execute Reusable State <i>Authorized</i> (local)</p> <p>2. Execute Reusable State <i>EnergyTransferStarted</i></p>	CSMS
Tool validations	<p>N/a</p> <p>Post scenario validations: N/a</p>	

Table 100. Test Case Id: TC_E_04_CS

Test case name	Local start transaction - Authorization first - Success	
Test case Id	TC_E_04_CS	
Use case Id(s)	E03 AND (C01 OR C02 OR C04 OR C06)	
Requirement(s)	E03.FR.01, E03.FR.06, E03.FR.12, E01.FR.16 AND C01.FR.02, C02.FR.01, C06.FR.02	
System under test	Charging Station	
Description	OCPP 2.x.x allows an EV driver to either first connect the EV and EVSE OR present a form of identification. Both sequences will result in being able to charge.	
Purpose	To verify if the Charging Station is able to start a charging session when the EV driver first presents a form of identification, before connecting the EV and EVSE.	
Prerequisite(s)	The Charging Station supports at least one authorization method described at the following Use cases; C01, C02, C04, C06.	
Before (Preparations)	Configuration State: <i>AuthCtrlr.AuthEnabled</i> is <i>true</i> (If implemented AND ReadWrite) <i>AuthCtrlr.DisableRemoteAuthorization</i> is <i>false</i> (If implemented)	
	Memory State: N/a	
	Reusable State(s): State is <i>ParkingBayOccupied</i> (Optional state)	
Main (Test scenario)	Charging Station	CSMS
	1. Execute Reusable State <i>Authorized</i> (local) 2. Execute Reusable State <i>EnergyTransferStarted</i>	
Tool validations	N/a Post scenario validations: N/a	

Table 101. Test Case Id: TC_E_05_CS

Test case name	Local start transaction - Authorization first - Cable plugin timeout	
Test case Id	TC_E_05_CS	
Use case Id(s)	E03 AND (C01 OR C02 OR C04 OR C06)	
Requirement(s)	E03.FR.01, E03.FR.05, E03.FR.06, E03.FR.12 AND C01.FR.02, C02.FR.01, C06.FR.02	
System under test	Charging Station	
Description	OCPP 2.x.x allows an EV driver to either first connect the EV and EVSE OR present a form of identification. Both sequences will result in being able to charge.	
Purpose	To verify if the Charging Station is able to deauthorize the transaction after the EVConnectionTimeout has expired.	
Prerequisite(s)	The Charging Station supports at least one authorization method described at the following Use cases; C01, C02, C04, C06.	
Before (Preparations)	<p>Configuration State:</p> <ul style="list-style-type: none"> - TxCtrlr.EVConnectionTimeOut is <Configured ev_connection_timeout> - AuthCtrlr.AuthEnabled is <i>true</i> (If implemented AND ReadWrite) - AuthCtrlr.DisableRemoteAuthorization is <i>false</i> (If implemented) - AuthCacheCtrlr.Enabled is <i>false</i> (If implemented) - AuthCtrlr.LocalPreAuthorize is <i>false</i> <p>Memory State: N/a</p> <p>Reusable State(s): State is Authorized (local)</p>	
Main (Test scenario)	<p>Charging Station</p> <p>1. The Charging Station sends a TransactionEventRequest</p> <p>Note(s):</p> <ul style="list-style-type: none"> - This step needs to be executed after the <Configured ev_connection_timeout> expires, if the transaction has been started. So in the case TxStartPoint contains ParkingBayOccupancy OR Authorized <p>Note(s):</p> <ul style="list-style-type: none"> - This step is only executed if TxStartPoint is ParkingBayOccupancy or Authorized - Optionally the Charging Station can send a StatusNotificationRequest or NotifyEventRequest with status Available <p>3. Execute Reusable State Authorized (local)</p> <p>Note(s):</p> <ul style="list-style-type: none"> - This step is executed to verify if the EVSE is actually ready to start another charging session. <p>4. Execute Reusable State EnergyTransferStarted</p>	CSMS
Tool validations	<ul style="list-style-type: none"> * Step 1: <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - triggerReason must be EVConnectTimeout <p>If <Configured TxStopPoint> contains Authorized then</p> <ul style="list-style-type: none"> eventType must be Ended AND transactionInfo.stoppedReason must be Timeout <p>Else eventType must be Updated</p> <p>Post scenario validations: N/a</p>	

Table 102. Test Case Id: TC_E_38_CS

Test case name	Local start transaction - EV not ready	
Test case Id	TC_E_38_CS	
Use case Id(s)	E03	
Requirement(s)	N/a	
System under test	Charging Station	
Description	OCPP 2.x.x allows an EV driver to either first connect the EV and EVSE OR present a form of identification. Both sequences will result in being able to charge.	
Purpose	To verify if the Charging Station is able to handle and report if an EV is not ready to start the energy transfer (yet).	
Prerequisite(s)	TxStartPoint should not be EnergyTransfer	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): State is <i>Authorized</i>	
Main (Test scenario)	Charging Station <u>Manual Action:</u> Set the EV to a state in which it is NOT ready for energy transfer. 1. Execute Reusable State <i>EVConnectedPreSession</i> 2. The Charging Station sends a TransactionEventRequest	CSMS 3. The OCTT responds with a TransactionEventResponse
Tool validations	* Step 2: Message: TransactionEventRequest - triggerReason must be <i>ChargingStateChanged</i> - transactionInfo.chargingState must be <i>SuspendedEV</i>	

Table 103. Test Case Id: TC_E_52_CS

Test case name	Local start transaction - Authorization first - DisableRemoteAuthorization	
Test case Id	TC_E_52_CS	
Use case Id(s)	E03 AND C01	
Requirement(s)	C01.FR.02, C01.FR.05,	
System under test	Charging Station	
Description	When DisableRemoteAuthorization is set to true, the Charging Station will only try to look up an IdToken in Authorization Cache or Local Authorization List, and not do an AuthorizeRequest for IdTokens. This overrules requirement C01.FR.02 and C01.FR.05.	
Purpose	To verify that the Charging Station will not send an AuthorizeRequest when DisableRemoteAuthorization is set to true.	
Prerequisite(s)	The Charging Station supports the authorization method described in C01. (RFID) AuthCtrlr.DisableRemoteAuthorization is implemented.	
Before (Preparations)	<p>Configuration State: AuthCtrlr.Enabled is true (If implemented) AuthCtrlr.DisableRemoteAuthorization is true</p> <p>Memory State: None of the configured valid IdTokens is present in Authorization Cache or Local Authorization List.</p> <p>Reusable State(s): State is ParkingBayOccupied (Optional state)</p>	
Main (Test scenario)	<p>Charging Station</p> <p>Manual Action: Present an idToken which is not configured in the Local Authorization List nor present in Authorization Cache.</p> <p>1. The Charging Station does NOT send a AuthorizeRequest</p>	CSMS
Tool validations	<ul style="list-style-type: none"> * Step 1: Check that Charging Station does NOT send an AuthorizeRequest and authorization is refused. <p>Post scenario validations: - N/a</p>	

Table 104. Test Case Id: TC_E_06_CS

Test case name	Local Stop Transaction - Accepted	
Test case Id	TC_E_06_CS	
Use case Id(s)	E07 AND (C01 OR C02 OR C04)	
Requirement(s)	E07.FR.04, E06.FR.15 AND C01.FR.03	
System under test	Charging Station	
Description	The EV Driver is able to stop an ongoing transaction, by locally presenting an IdToken.	
Purpose	To verify whether the Charging Station is able to perform a local stop authorization.	
Prerequisite(s)	The Charging Station supports at least one authorization method described at the following Use cases; C01, C02, C04.	
Before (Preparations)	<p>Configuration State: AuthCtrlr.AuthEnabled is <i>true</i> (If implemented AND ReadWrite) AuthCtrlr.DisableRemoteAuthorization is <i>false</i> (If implemented)</p> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>EnergyTransferStarted</i></p>	
Main (Test scenario)	Charging Station	CSMS
	1. Execute Reusable State <i>StopAuthorized</i> (local) 2. Execute Reusable State <i>EVConnectedPostSession</i> 3. Execute Reusable State <i>EVDDisconnected</i> 4. Execute Reusable State <i>ParkingBayUnoccupied</i>	
Tool validations	N/a Post scenario validations: N/a	

Table 105. Test Case Id: TC_E_07_CS

Test case name	Stop transaction options - PowerPathClosed - Local stop	
Test case Id	TC_E_07_CS	
Use case Id(s)	E06(S5)	
Requirement(s)	E06.FR.06, E06.FR.15	
System under test	Charging Station	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the stop options.	
Purpose	To verify if the Charging Station stops a transaction when it is locally stopped by an EV driver and TxStopPoint contains <i>PowerPathClosed</i> .	
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station does NOT have the following configuration; The mutability of TxStopPoint is <i>ReadOnly</i> AND (the value <i>PowerPathClosed</i> is NOT set OR <i>Authorized</i> is set). - If the mutability of TxStopPoint is <i>ReadWrite</i>, then the value <i>PowerPathClosed</i> must be supported. 	
Before (Preparations)	<p>Configuration State: TxStopPoint contains <i>PowerPathClosed</i></p> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>EnergyTransferStarted</i></p>	
Main (Test scenario)	Charging Station	CSMS
	<p><u>Manual Action:</u> Present <i>IdToken</i> to stop charging session.</p> <p>1. Execute Reusable State <i>StopAuthorized</i></p>	
Tool validations	<p>N/a</p> <p>Post scenario validations: N/a</p>	

Table 106. Test Case Id: TC_E_35_CS

Test case name	Stop transaction options - PowerPathClosed - Remote stop	
Test case Id	TC_E_35_CS	
Use case Id(s)	E06(S5)	
Requirement(s)	E06.FR.06, E06.FR.15	
System under test	Charging Station	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the stop options.	
Purpose	To verify if the Charging Station stops a transaction when it is remotely stopped the CSMS and TxStopPoint contains PowerPathClosed .	
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station does NOT have the following configuration; The mutability of TxStopPoint is ReadOnly AND (the value PowerPathClosed is NOT set OR Authorized is set). - If the mutability of TxStopPoint is ReadWrite, then the value PowerPathClosed must be supported. 	
Before (Preparations)	<p>Configuration State: TxStopPoint contains <i>PowerPathClosed</i></p> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>EnergyTransferStarted</i></p>	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with a RequestStopTransactionResponse	1. The OCTT sends a RequestStopTransactionRequest with transactionId <transactionId provided by the Charging Station in TransactionEventRequest >
	3. The Charging Station sends a TransactionEventRequest	4. The OCTT responds with a TransactionEventResponse
Tool validations	<ul style="list-style-type: none"> * Step 2: Message: RequestStopTransactionResponse - status must be <i>Accepted</i> * Step 3: Message: TransactionEventRequest - triggerReason must be <i>RemoteStop</i> - transactionInfo.stoppedReason must be <i>Remote</i> - eventType must be <i>Ended</i> <p>Post scenario validations: N/a</p>	

Table 107. Test Case Id: TC_E_37_CS

Test case name	Stop transaction options - PowerPathClosed - EV side disconnect			
Test case Id	TC_E_37_CS			
Use case Id(s)	E06(S5)			
Requirement(s)	E06.FR.06, E06.FR.15			
System under test	Charging Station			
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the stop options.			
Purpose	To verify if the Charging Station stops a transaction when the EV and the EVSE get disconnected and TxStopPoint contains <i>PowerPathClosed</i> .			
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station does NOT have the following configuration; The mutability of TxStopPoint is <i>ReadOnly</i> AND (the value <i>PowerPathClosed</i> is NOT set OR (<i>EnergyTransfer</i> OR <i>EVConnected</i> OR <i>DataSigned</i> is set)). - If the mutability of TxStopPoint is <i>ReadWrite</i>, then the value <i>PowerPathClosed</i> must be supported. 			
Before (Preparations)	<p>Configuration State: TxStopPoint contains <i>PowerPathClosed</i> StopTxOnEVSideDisconnect is <i>false</i> (If mutability is <i>ReadWrite</i>)</p> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>EnergyTransferSuspended</i></p>			
Main (Scenario)	<p>Charging Station</p> <p>Manual Action: Disconnect the EV and EVSE on EV side (EVSE loses connection with EV).</p> <table border="1"> <tr> <td>1. The Charging Station sends a TransactionEventRequest</td> <td>2. The OCTT responds with a TransactionEventResponse</td> </tr> </table>	1. The Charging Station sends a TransactionEventRequest	2. The OCTT responds with a TransactionEventResponse	CSMS
1. The Charging Station sends a TransactionEventRequest	2. The OCTT responds with a TransactionEventResponse			
Tool validations	<p>* Step 1: Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - triggerReason must be <i>EVCommunicationLost</i> - transactionInfo.chargingState must be <i>Idle</i> - transactionInfo.stoppedReason must be <i>EVDisconnected</i> - eventType must be <i>Ended</i> <p>Post scenario validations: N/a</p>			

Table 108. Test Case Id: TC_E_08_CS

Test case name	Stop transaction options - EnergyTransfer stopped - StopAuthorized	
Test case Id	TC_E_08_CS	
Use case Id(s)	E06(S6)	
Requirement(s)	E06.FR.07, E06.FR.15	
System under test	Charging Station	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the stop options.	
Purpose	To verify if the Charging Station stops a transaction when the energy transfer stopped normally and it has been configured to do so.	
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station does NOT have the following configuration; The mutability of TxStopPoint is ReadOnly AND (the value <i>EnergyTransfer</i> is NOT set OR (Authorized OR PowerPathClosed) is set). - If the mutability of TxStopPoint is ReadWrite, then the value <i>EnergyTransfer</i> must be supported. 	
Before (Preparations)	<p>Configuration State: TxStopPoint contains <i>EnergyTransfer</i></p> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>EnergyTransferStarted</i></p>	
Main (Test scenario)	Charging Station	CSMS
	1. State is <i>StopAuthorized</i>	
Tool validations	<ul style="list-style-type: none"> * Step 1: N/a <p>Post scenario validations: N/a</p>	

Table 109. Test Case Id: TC_E_22_CS

Test case name	Stop transaction options - EnergyTransfer stopped - SuspendedEV	
Test case Id	TC_E_22_CS	
Use case Id(s)	E06(S6)	
Requirement(s)	E06.FR.07, E06.FR.15	
System under test	Charging Station	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the stop options.	
Purpose	To verify if the Charging Station stops a transaction when the energy transfer stopped by the EV and the Charging Station has been configured to do so.	
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station does NOT have the following configuration; The mutability of TxStopPoint is ReadOnly AND the value <i>EnergyTransfer</i> is NOT set. - If the mutability of TxStopPoint is ReadWrite, then the value <i>EnergyTransfer</i> must be supported. 	
Before (Preparations)	<p>Configuration State: TxStopPoint contains <i>EnergyTransfer</i></p> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>EnergyTransferStarted</i></p>	
Main (Test scenario)	<p>Charging Station</p> <p>Manual Action: The EV suspends the energy transfer.</p>	CSMS
	<p>1. The Charging Station sends a TransactionEventRequest</p>	<p>2. The OCTT responds with a TransactionEventResponse</p>
Tool validations	<p>* Step 1:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - triggerReason must be <i>ChargingStateChanged</i> - transactionInfo.chargingState must be <i>EVConnected</i> + OR - transactionInfo.chargingState must be <i>SuspendedEV</i> AND - transactionInfo.stoppedReason must be <i>StoppedByEV</i> - eventType must be <i>Ended</i> (if chargingState is <i>EVConnected</i>) OR - eventType must be <i>Updated</i> (if chargingState is <i>SuspendedEV</i>) <p>Post scenario validations: N/a</p>	

Table 110. Test Case Id: TC_E_14_CS

Test case name	Stop transaction options - EVDisconnected - Charging Station side	
Test case Id	TC_E_14_CS	
Use case Id(s)	E06(S2)	
Requirement(s)	E06.FR.02, E06.FR.15	
System under test	Charging Station	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the stop options.	
Purpose	To verify if the Charging Station stops a transaction when the EV and EVSE are disconnected at the Charging Station side and it has been configured to do so.	
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station does NOT have the following configuration; The mutability of TxStopPoint is <i>ReadOnly</i> AND (the value EVConnected is NOT set OR (<i>EnergyTransfer</i> OR <i>PowerPathClosed</i> OR <i>DataSigned</i> OR <i>Authorized</i> is set)). - If the mutability of TxStopPoint is <i>ReadWrite</i>, then the value EVConnected must be supported. 	
Before (Preparations)	<p>Configuration State: TxStopPoint contains <i>EVConnected</i></p> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>EVConnectedPostSession</i></p>	
Main (Scenario)	<p>Charging Station</p> <p><u>Manual Action:</u> Disconnect the EV and EVSE.</p> <p>1. The Charging Station notifies the CSMS about the status change of the connector.</p> <p>3. The Charging Station sends a TransactionEventRequest</p>	<p>CSMS</p> <p>2. The OCTT responds accordingly.</p> <p>4. The OCTT responds with a TransactionEventResponse</p>
Tool validations	<p>* Step 1: Message: StatusNotificationRequest - connectorStatus must be <i>Available</i></p> <p>Message: NotifyEventRequest - eventData[0].trigger must be <i>Delta</i> - eventData[0].actualValue must be <i>Available</i> - eventData[0].component.name must be <i>Connector</i> - eventData[0].variable.name must be <i>AvailabilityState</i></p> <p>* Step 3: Message: TransactionEventRequest - triggerReason must be <i>EVCommunicationLost</i> - transactionInfo.chargingState must be <i>Idle</i> - If the OCTT is configured to stop transactions using a RequestStopTransactionRequest message then transactionInfo.stoppedReason must be <i>Remote</i> Else transactionInfo.stoppedReason must be <i>Local</i> or <i>EVDisconnected</i> - eventType must be <i>Ended</i></p> <p>Post scenario validations: N/a</p>	

Table 111. Test Case Id: TC_E_20_CS

Test case name	Stop transaction options - EVDisconnected - EV side (able to charge IEC 61851-1 EV)	
Test case Id	TC_E_20_CS	
Use case Id(s)	E06(S2), E10	
Requirement(s)	E06.FR.02, E06.FR.15	
System under test	Charging Station	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the stop options.	
Purpose	To verify if the Charging Station stops a transaction when the EV and EVSE are disconnected at the EV side and it has been configured to do so.	
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station does NOT have the following configuration; The mutability of TxStopPoint is <i>ReadOnly</i> AND (the value <i>EVConnected</i> is NOT set OR (<i>EnergyTransfer</i> OR <i>PowerPathClosed</i> is set)). - If the mutability of TxStopPoint is <i>ReadWrite</i>, then the value <i>EVConnected</i> must be supported. - The Charging Station does NOT have following configuration combination; StopTxOnEVSideDisconnect mutability <i>ReadOnly</i> with value <i>true</i> AND TxStopPoint mutability is <i>ReadOnly</i> and contains <i>Authorized</i>. - The Charging Station is able to charge with a EV that uses IEC 61851-1. 	
Before (Preparations)	<p>Configuration State: TxStopPoint contains <i>EVConnected</i> StopTxOnEVSideDisconnect is <i>false</i> (If mutability is <i>ReadWrite</i>)</p> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>EnergyTransferSuspended</i></p>	
Main (Scenario)	<p>Charging Station</p> <p>CSMS</p> <p>Manual Action: Disconnect the EV and EVSE on EV side (EVSE loses connection with EV).</p>	<p>1. The Charging Station sends a TransactionEventRequest</p> <p>2. The OCTT responds with a TransactionEventResponse</p>
Tool validations	<p>* Step 1: Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - triggerReason must be <i>EVCommunicationLost</i> - transactionInfo.chargingState must be <i>Idle</i> - transactionInfo.stoppedReason must be <i>EVDisconnected</i> - eventType must be <i>Ended</i> <p>Post scenario validations: N/a</p>	

Table 112. Test Case Id: TC_E_54_CS

Test case name	Stop transaction options - EVDisconnected - EV side (not able to charge IEC 61851-1 EV)			
Test case Id	TC_E_54_CS			
Use case Id(s)	E06(S2), E10			
Requirement(s)	E06.FR.02, E06.FR.15			
System under test	Charging Station			
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the stop options.			
Purpose	To verify if the Charging Station stops a transaction when the EV and EVSE are disconnected at the EV side and it has been configured to do so.			
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station does NOT have the following configuration; The mutability of TxStopPoint is <i>ReadOnly</i> AND (the value <i>EVConnected</i> is NOT set OR (<i>EnergyTransfer</i> OR <i>PowerPathClosed</i> is set)). - If the mutability of TxStopPoint is <i>ReadWrite</i>, then the value <i>EVConnected</i> must be supported. - The Charging Station does NOT have following configuration combination; StopTxOnEVSideDisconnect mutability <i>ReadOnly</i> with value <i>true</i> AND TxStopPoint mutability is <i>ReadOnly</i> and contains <i>Authorized</i>. - The Charging Station supports high level communication. 			
Before (Preparations)	<p>Configuration State: TxStopPoint contains <i>EVConnected</i> StopTxOnEVSideDisconnect is <i>false</i> (If mutability is <i>ReadWrite</i>)</p> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>EnergyTransferSuspended</i></p>			
Main (Scenario)	<p>Charging Station</p> <p>Manual Action: Disconnect the EV and EVSE on EV side (EVSE loses connection with EV).</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="padding: 5px;">1. The Charging Station sends a TransactionEventRequest</td> <td style="padding: 5px;">2. The OCTT responds with a TransactionEventResponse</td> </tr> </table>	1. The Charging Station sends a TransactionEventRequest	2. The OCTT responds with a TransactionEventResponse	<p>CSMS</p>
1. The Charging Station sends a TransactionEventRequest	2. The OCTT responds with a TransactionEventResponse			
Tool validations	<p>* Step 1: Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - triggerReason must be <i>EVCommunicationLost</i> - transactionInfo.chargingState must be <i>Idle</i> - transactionInfo.stoppedReason must be <i>StoppedByEV</i> or <i>EVDisconnected</i> - eventType must be <i>Ended</i> <p>Post scenario validations: N/a</p>			

Table 113. Test Case Id: TC_E_15_CS

Test case name	Stop transaction options - StopAuthorized - Local	
Test case Id	TC_E_15_CS	
Use case Id(s)	E06(S3)	
Requirement(s)	E06.FR.03, E06.FR.15	
System under test	Charging Station	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the stop options.	
Purpose	To verify if the Charging Station stops a transaction when the EV driver locally stops the transaction and it has been configured to do so.	
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station supports at least one authorization method described at the following Use cases; C01, C02, C04. - The Charging Station does NOT have the following configuration; The mutability of TxStopPoint is ReadOnly AND the value Authorized is NOT set OR PowerPathClosed is set. - If the mutability of TxStopPoint is ReadWrite, then the value Authorized must be supported. 	
Before (Preparations)	<p>Configuration State: TxStopPoint contains Authorized</p> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>EnergyTransferStarted</i></p>	
Main (Test scenario)	<p>Charging Station</p> <p><u>Notes(s):</u> The tool will wait for <Configured Transaction Duration> seconds</p> <p><u>Manual Action:</u> Present IdToken to stop charging session.</p>	<p>CSMS</p> <p>1. The Charging Station sends a TransactionEventRequest</p> <p>2. The OCTT responds with a TransactionEventResponse</p>
Tool validations	<p>* Step 1: Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - triggerReason must be StopAuthorized - transactionInfo.stoppedReason must be Local - eventType must be Ended <p>Post scenario validations: N/a</p>	

Table 114. Test Case Id: TC_E_21_CS

Test case name	Stop transaction options - StopAuthorized - Remote	
Test case Id	TC_E_21_CS	
Use case Id(s)	E06(S3) AND F03	
Requirement(s)	E06.FR.03, E06.FR.15 AND F03.FR.09	
System under test	Charging Station	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the stop options.	
Purpose	To verify if the Charging Station stops a transaction when it receives a RequestStopTransactionRequest and it has been configured to do so.	
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station does NOT have the following configuration; The mutability of TxStopPoint is ReadOnly AND the value Authorized is NOT set OR PowerPathClosed is set. - If the mutability of TxStopPoint is ReadWrite, then the value Authorized must be supported. 	
Before (Preparations)	<p>Configuration State: TxStopPoint contains Authorized</p> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>EnergyTransferStarted</i></p>	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with a RequestStopTransactionResponse	1. The OCTT sends a RequestStopTransactionRequest with transactionId <transactionId provided by the Charging Station in TransactionEventRequest >
	3. The Charging Station sends a TransactionEventRequest	4. The OCTT responds with a TransactionEventResponse
Tool validations	<ul style="list-style-type: none"> * Step 2: Message: RequestStopTransactionResponse - status must be <i>Accepted</i> * Step 3: Message: TransactionEventRequest - triggerReason must be <i>RemoteStop</i> - transactionInfo.stoppedReason must be <i>Remote</i> - eventType must be <i>Ended</i> <p>Post scenario validations: N/a</p>	

Table 115. Test Case Id: TC_E_19_CS

Test case name	Stop transaction options - ParkingBayUnoccupied	
Test case Id	TC_E_19_CS	
Use case Id(s)	E06(S1)	
Requirement(s)	E06.FR.01, E06.FR.15	
System under test	Charging Station	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the stop options.	
Purpose	To verify if the Charging Station stops a transaction when the EV left the parking bay and it has been configured to do so.	
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station does NOT have the following configuration; The mutability of TxStopPoint is <i>ReadOnly</i> AND (the value <i>ParkingBayOccupied</i> is NOT set OR (<i>EnergyTransfer</i> OR <i>PowerPathClosed</i> OR <i>DataSigned</i> OR <i>Authorized</i> OR <i>EVConnected</i> is set)). - If the mutability of TxStopPoint is <i>ReadWrite</i>, then the value <i>ParkingBayOccupied</i> must be supported. 	
Before (Preparations)	<p>Configuration State: TxStopPoint contains <i>ParkingBayOccupied</i></p> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>EVDisconnected</i></p>	
Main (Scenario)	<p>Charging Station</p> <p>Manual Action: Drive EV out of parking bay.</p>	<p>CSMS</p> <p>1. The Charging Station sends a TransactionEventRequest</p> <p>2. The OCTT responds with a TransactionEventResponse</p>
Tool validations	<p>* Step 1: Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - triggerReason must be <i>EVDeparted</i> - If the OCTT is configured to stop transactions using a RequestStopTransactionRequest message then transactionInfo.stoppedReason must be <i>Remote</i> Else transactionInfo.stoppedReason must be <i>Local</i> - eventType must be <i>Ended</i> <p>Post scenario validations: N/a</p>	

Table 116. Test Case Id: TC_E_24_CS

Test case name	Disconnect cable on EV-side - Deauthorize transaction - UnlockOnEVSideDisconnect is true											
Test case Id	TC_E_24_CS											
Use case Id(s)	E09											
Requirement(s)	E09.FR.01, E09.FR.02, E09.FR.04											
System under test	Charging Station											
Description	The Charging Station can behave in several different ways when the cable is disconnected at the EV side, based on its configuration. This test case tests one of the possible configuration settings.											
Purpose	To verify if the Charging Station deauthorizes the transaction when the EV and EVSE are disconnected at the EV side and it has been configured to do so AND unlocks the cable at Charging Station side.											
Prerequisite(s)	The Charging Station does NOT have a permanently attached cable.											
Before (Preparations)	<p>Configuration State: StopTxOnEVSideDisconnect is true UnlockOnEVSideDisconnect is true</p> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>EnergyTransferSuspended</i></p>											
Main (Scenario)	<table border="1"> <tr> <td>Charging Station</td> <td>CSMS</td> </tr> <tr> <td colspan="2"><u>Manual Action: Disconnect the EV and EVSE on EV side (EVSE loses connection with EV).</u></td> </tr> <tr> <td>1. The Charging Station sends a TransactionEventRequest</td> <td>2. The OCTT responds with a TransactionEventResponse</td> </tr> <tr> <td colspan="2"><u>Manual Action: Disconnect the EV and EVSE on Charging Station side.</u></td> </tr> <tr> <td>3. The Charging Station notifies the CSMS about the current state of the connector.</td> <td>4. The OCTT responds accordingly.</td> </tr> </table>	Charging Station	CSMS	<u>Manual Action: Disconnect the EV and EVSE on EV side (EVSE loses connection with EV).</u>		1. The Charging Station sends a TransactionEventRequest	2. The OCTT responds with a TransactionEventResponse	<u>Manual Action: Disconnect the EV and EVSE on Charging Station side.</u>		3. The Charging Station notifies the CSMS about the current state of the connector.	4. The OCTT responds accordingly.	
Charging Station	CSMS											
<u>Manual Action: Disconnect the EV and EVSE on EV side (EVSE loses connection with EV).</u>												
1. The Charging Station sends a TransactionEventRequest	2. The OCTT responds with a TransactionEventResponse											
<u>Manual Action: Disconnect the EV and EVSE on Charging Station side.</u>												
3. The Charging Station notifies the CSMS about the current state of the connector.	4. The OCTT responds accordingly.											
Tool validations	<ul style="list-style-type: none"> * Step 1: Message: TransactionEventRequest - triggerReason must be <i>EVCommunicationLost</i> - transactionInfo.chargingState must be <i>Idle</i> * Step 3: Message: StatusNotificationRequest - connectorStatus <i>Available</i> <p>Message: NotifyEventRequest - eventData[0].trigger Delta - eventData[0].actualValue "Available" - eventData[0].component.name "Connector" - eventData[0].variable.name "AvailabilityState"</p> <p>Post scenario validations: N/a</p>											

Table 117. Test Case Id: TC_E_25_CS

Test case name	Disconnect cable on EV-side - Deauthorize transaction - UnlockOnEVSideDisconnect is false							
Test case Id	TC_E_25_CS							
Use case Id(s)	E09							
Requirement(s)	E09.FR.01, E09.FR.03, E09.FR.04							
System under test	Charging Station							
Description	The Charging Station can behave in several different ways when the cable is disconnected at the EV side, based on its configuration. This test case tests one of the possible configuration settings.							
Purpose	To verify if the Charging Station deauthorizes the transaction when the EV and EVSE are disconnected at the EV side and it has been configured to do so AND keeps the cable locked at Charging Station side.							
Prerequisite(s)	N/a							
Before (Preparations)	<p>Configuration State: StopTxOnEVSideDisconnect is true UnlockOnEVSideDisconnect is false</p> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>EnergyTransferSuspended</i></p>							
Main (Scenario)	<table border="1"> <tr> <td>Charging Station</td> <td>CSMS</td> </tr> <tr> <td colspan="2"><u>Manual Action:</u> Disconnect the EV and EVSE on EV side (EVSE loses connection with EV).</td> </tr> <tr> <td>1. The Charging Station sends a TransactionEventRequest</td> <td>2. The OCTT responds with a TransactionEventResponse</td> </tr> </table> <p><u>Manual Action:</u> Present the <i>IdToken</i> that was used to start the transaction.</p> <p><u>Note(s):</u> - This manual action needs to be executed when the Charging Station has a detachable cable on the Charging Station side.</p> <p><u>Manual Action:</u> Disconnect the EV and EVSE on Charging Station side.</p> <p><u>Note(s):</u> - This manual action needs to be executed when the Charging Station has a detachable cable on the Charging Station side.</p> <p>3. The Charging Station notifies the CSMS about the current state of the connector.</p>	Charging Station	CSMS	<u>Manual Action:</u> Disconnect the EV and EVSE on EV side (EVSE loses connection with EV).		1. The Charging Station sends a TransactionEventRequest	2. The OCTT responds with a TransactionEventResponse	<p>CSMS</p> <p>2. The OCTT responds with a TransactionEventResponse</p> <p>4. The OCTT responds accordingly.</p>
Charging Station	CSMS							
<u>Manual Action:</u> Disconnect the EV and EVSE on EV side (EVSE loses connection with EV).								
1. The Charging Station sends a TransactionEventRequest	2. The OCTT responds with a TransactionEventResponse							
Tool validations	<ul style="list-style-type: none"> * Step 1: Message: TransactionEventRequest - triggerReason must be <i>EVCommunicationLost</i> - transactionInfo.chargingState must be <i>Idle</i> * Step 3: Message: StatusNotificationRequest - connectorStatus <i>Available</i> Message: NotifyEventRequest - eventData[0].trigger <i>Delta</i> - eventData[0].actualValue "Available" - eventData[0].component.name "Connector" - eventData[0].variable.name "AvailabilityState" <p>Post scenario validations: N/a</p>							

Table 118. Test Case Id: TC_E_26_CS

Test case name	Disconnect cable on EV-side - Suspend transaction																		
Test case Id	TC_E_26_CS																		
Use case Id(s)	E10																		
Requirement(s)	E10.FR.01, E10.FR.03																		
System under test	Charging Station																		
Description	The Charging Station can behave in several different ways when the cable is disconnected at the EV side, based on its configuration. This test case tests one of the possible configuration settings.																		
Purpose	To verify if the Charging Station suspends the transaction when the EV and EVSE are disconnected at the EV side and it has been configured to do so AND is able restart the energy transfer after reconnecting the EV and EVSE.																		
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station does NOT have the following configuration; The mutability of TxStopPoint is <i>ReadOnly</i> AND (the value <i>Authorized</i> OR <i>ParkingBayOccupancy</i> is NOT set OR (<i>EnergyTransfer</i> OR <i>PowerPathClosed</i> OR <i>DataSigned</i> OR <i>EVConnected</i> is set)). - If the mutability of TxStopPoint is <i>ReadWrite</i>, then the value <i>Authorized</i> OR <i>ParkingBayOccupancy</i> must be supported. 																		
Before (Preparations)	<p>Configuration State: TxStopPoint contains <i>Authorized</i> (If supported) AND/OR <i>ParkingBayOccupancy</i> (If supported) UnlockOnEVSideDisconnect is <i>false</i> StopTxOnEVSideDisconnect is <i>false</i></p> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>EnergyTransferSuspended</i></p>																		
Main (Scenario)	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="padding: 5px;">Charging Station</td> <td style="padding: 5px;">CSMS</td> </tr> <tr> <td colspan="2" style="padding: 5px;"><u>Manual Action:</u> Disconnect the EV and EVSE on EV side (EVSE loses connection with EV).</td> </tr> <tr> <td style="padding: 5px;">1. The Charging Station sends a TransactionEventRequest</td> <td style="padding: 5px;">2. The OCTT responds with a TransactionEventResponse</td> </tr> <tr> <td style="padding: 5px;">3. The Charging Station notifies the CSMS about the current state of the connector.</td> <td style="padding: 5px;">4. The OCTT responds accordingly.</td> </tr> <tr> <td colspan="2" style="padding: 5px;"><u>Note(s):</u> - This step needs to be executed when the Charging Station has a permanently attached cable on the Charging Station side.</td> </tr> <tr> <td colspan="2" style="padding: 5px;"><u>Manual Action:</u> Reconnect the EV and EVSE on EV side.</td> </tr> <tr> <td colspan="2" style="padding: 5px;"><u>Note(s):</u> - If the Charging Station has a permanently attached cable on the Charging Station side, then this step needs to be executed before the configured EVConnectionTimeout expires.</td> </tr> <tr> <td style="padding: 5px;">5. The Charging Station sends a TransactionEventRequest</td> <td style="padding: 5px;">6. The OCTT responds with a TransactionEventResponse</td> </tr> <tr> <td style="padding: 5px;">7. The Charging Station sends a TransactionEventRequest</td> <td style="padding: 5px;">8. The OCTT responds with a TransactionEventResponse</td> </tr> </table>	Charging Station	CSMS	<u>Manual Action:</u> Disconnect the EV and EVSE on EV side (EVSE loses connection with EV).		1. The Charging Station sends a TransactionEventRequest	2. The OCTT responds with a TransactionEventResponse	3. The Charging Station notifies the CSMS about the current state of the connector.	4. The OCTT responds accordingly.	<u>Note(s):</u> - This step needs to be executed when the Charging Station has a permanently attached cable on the Charging Station side.		<u>Manual Action:</u> Reconnect the EV and EVSE on EV side.		<u>Note(s):</u> - If the Charging Station has a permanently attached cable on the Charging Station side, then this step needs to be executed before the configured EVConnectionTimeout expires.		5. The Charging Station sends a TransactionEventRequest	6. The OCTT responds with a TransactionEventResponse	7. The Charging Station sends a TransactionEventRequest	8. The OCTT responds with a TransactionEventResponse
Charging Station	CSMS																		
<u>Manual Action:</u> Disconnect the EV and EVSE on EV side (EVSE loses connection with EV).																			
1. The Charging Station sends a TransactionEventRequest	2. The OCTT responds with a TransactionEventResponse																		
3. The Charging Station notifies the CSMS about the current state of the connector.	4. The OCTT responds accordingly.																		
<u>Note(s):</u> - This step needs to be executed when the Charging Station has a permanently attached cable on the Charging Station side.																			
<u>Manual Action:</u> Reconnect the EV and EVSE on EV side.																			
<u>Note(s):</u> - If the Charging Station has a permanently attached cable on the Charging Station side, then this step needs to be executed before the configured EVConnectionTimeout expires.																			
5. The Charging Station sends a TransactionEventRequest	6. The OCTT responds with a TransactionEventResponse																		
7. The Charging Station sends a TransactionEventRequest	8. The OCTT responds with a TransactionEventResponse																		

Test case name	Disconnect cable on EV-side - Suspend transaction
Tool validations	<ul style="list-style-type: none"> * Step 1: Message: TransactionEventRequest - triggerReason must be <i>EVCommunicationLost</i> - transactionInfo.chargingState must be <i>Idle</i> - eventType must be <i>Updated</i> * Step 3: Message: StatusNotificationRequest - connectorStatus <i>Available</i> Message: NotifyEventRequest - eventData[0].trigger <i>Delta</i> - eventData[0].actualValue "<i>Available</i>" - eventData[0].component.name "<i>Connector</i>" - eventData[0].variable.name "<i>AvailabilityState</i>" * Step 5: Message: TransactionEventRequest - triggerReason must be <i>CablePluggedIn</i> - transactionInfo.chargingState must be <i>EVConnected</i> - eventType must be <i>Updated</i> * Step 7: Message: TransactionEventRequest - triggerReason must be <i>ChargingStateChanged</i> - transactionInfo.chargingState must be <i>Charging</i> - eventType must be <i>Updated</i> <p>Post scenario validations: N/a</p>

Table 119. Test Case Id: TC_E_27_CS

Test case name	Disconnect cable on EV-side - Suspend transaction - Fixed cable connection timeout	
Test case Id	TC_E_27_CS	
Use case Id(s)	E10	
Requirement(s)	E10.FR.02, E10.FR.03	
System under test	Charging Station	
Description	The Charging Station can behave in several different ways when the cable is disconnected at the EV side, based on its configuration. This test case tests one of the possible configuration settings.	
Purpose	To verify if the Charging Station suspends the transaction when the EV and EVSE are disconnected at the EV side and it has been configured to do so AND deauthorizes the transaction after the configured connection timeout expires.	
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station does NOT have the following configuration; The mutability of TxStopPoint is <i>ReadOnly</i> AND (the value <i>Authorized</i> OR <i>ParkingBayOccupancy</i> is NOT set OR (<i>EnergyTransfer</i> OR <i>PowerPathClosed</i> OR <i>DataSigned</i> OR <i>EVConnected</i> is set)). - If the mutability of TxStopPoint is <i>_ReadWrite</i>, then the value <i>Authorized</i> OR <i>ParkingBayOccupancy</i> must be supported. - The Charging Station has a permanently attached cable at the Charging Station side. 	
Before (Preparations)	<p>Configuration State:</p> <p>TxStopPoint contains <i>Authorized</i> (If supported)</p> <p>TxStopPoint contains <i>ParkingBayOccupancy</i> (If supported)</p> <p>UnlockOnEVSideDisconnect is <i>false</i></p> <p>StopTxOnEVSideDisconnect is <i>false</i></p> <p>Memory State:</p> <p>N/a</p> <p>Reusable State(s):</p> <p>State is <i>EnergyTransferSuspended</i></p>	
Main (Scenario)	<p>Charging Station</p> <p><u>Manual Action:</u> Disconnect the EV and EVSE on EV side (EVSE loses connection with EV).</p> <p>1. The Charging Station sends a TransactionEventRequest</p> <p>3. The Charging Station notifies the CSMS about the current state of the connector.</p> <p><u>Note(s):</u></p> <ul style="list-style-type: none"> - This step needs to be executed when the Charging Station has a permanently attached cable on the Charging Station side. <p><u>Manual Action:</u> Reconnect the EV and EVSE on EV side.</p> <p><u>Note(s):</u></p> <ul style="list-style-type: none"> - If the Charging Station has a permanently attached cable on the Charging Station side, then this step needs to be executed before the configured EVConnectionTimeout expires. <p>5. The Charging Station sends a TransactionEventRequest</p>	<p>CSMS</p> <p>2. The OCTT responds with a TransactionEventResponse</p> <p>4. The OCTT responds accordingly.</p> <p>6. The OCTT responds with a TransactionEventResponse</p> <p><u>Note(s):</u> Optionally the Charging Station can send a <i>StatusNotificationRequest</i> or <i>NotifyEventRequest</i> with status Available</p>

Test case name	Disconnect cable on EV-side - Suspend transaction - Fixed cable connection timeout
Tool validations	<ul style="list-style-type: none"> * Step 1: Message: TransactionEventRequest - triggerReason must be <i>EVCommunicationLost</i> - transactionInfo.chargingState must be <i>Idle</i> - eventType must be <i>Updated</i> * Step 3: Message: StatusNotificationRequest - connectorStatus <i>Available</i> Message: NotifyEventRequest - eventData[0].trigger <i>Delta</i> - eventData[0].actualValue "<i>Available</i>" - eventData[0].component.name "<i>Connector</i>" - eventData[0].variable.name "<i>AvailabilityState</i>" * Step 5: Message: TransactionEventRequest - triggerReason must be <i>EVConnectTimeout</i> If <Configured TxCtrlr.TxStopPoint> contains <i>Authorized</i> then eventType must be <i>Ended</i> transactionInfo.stoppedReason must be <i>Timeout</i> else if <Configured TxCtrlr.TxStopPoint> contains <i>ParkingBayOccupancy</i> then eventType must be <i>Updated</i>
	Post scenario validations: N/a

Table 120. Test Case Id: TC_E_28_CS

Test case name	Check Transaction status - TransactionId unknown	
Test case Id	TC_E_28_CS	
Use case Id(s)	E14	
Requirement(s)	E14.FR.01	
System under test	Charging Station	
Description	The CSMS is able to request the status of a transaction and to find out whether there are queued transaction-related messages, using the GetTransactionStatusRequest message.	
Purpose	To verify if the Charging Station is able to handle receiving a GetTransactionStatusRequest for an unknown transactionId.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station 2. The Charging Station responds with a GetTransactionStatusResponse	CSMS 1. The OCTT sends a GetTransactionStatusRequest with transactionId <Randomly generated transactionId>
Tool validations	* Step 2: Message: GetTransactionStatusResponse - ongoingIndicator must be <i>false</i> - messagesInQueue must be <i>false</i> Post scenario validations: N/a	

Table 121. Test Case Id: TC_E_29_CS

Test case name	Check Transaction status - Transaction with id ongoing - with message in queue	
Test case Id	TC_E_29_CS	
Use case Id(s)	E14	
Requirement(s)	E14.FR.02,E14.FR.04	
System under test	Charging Station	
Description	The CSMS is able to request the status of a transaction and to find out whether there are queued transaction-related messages, using the GetTransactionStatusRequest message.	
Purpose	To verify if the Charging Station is able to correctly respond to a GetTransactionStatusRequest with a transactionId, while there is a message queued belonging to an ongoing transaction with the requested id.	
Prerequisite(s)	N/a	
Before (Preparations)	<p>Configuration State:</p> <p>SampledDataTxUpdatedMeasurands is <Configured transaction_updated_metervalues_measurands></p> <p>SampledDataTxUpdatedInterval is <Configured transaction_updated_metervalues_interval></p> <p>OfflineThreshold is <Configured RetryBackOffWaitMinimum_duration> + 60.0</p> <p>RetryBackOffWaitMinimum is <Configured RetryBackOffWaitMinimum_duration></p> <p>RetryBackOffRandomRange is 0</p> <p>Note: <Configured RetryBackOffWaitMinimum_duration> must be greater than <Configured Transaction MeterValues interval></p> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>EnergyTransferStarted</i></p>	
Main (Test scenario)	<p>Charging Station</p> <p>4. The Charging Station responds with a GetTransactionStatusResponse</p> <p>Note(s): - The Charging Station will empty its Transaction message queue. This will contain one or more TransactionEventRequest messages</p>	<p>CSMS</p> <p>1. The OCTT closes the WebSocket connection AND does not accept a reconnect.</p> <p>2. The OCTT waits a number of seconds equal to <Configured RetryBackOffWaitMinimum_duration>, before accepting a reconnection attempt from the Charging Station.</p> <p>3. The OCTT sends a GetTransactionStatusRequest with transactionId <Generated transactionId from Before></p> <p>Note: <i>This step will be executed the moment the WebSocket connection is restored.</i></p> <p>5. The Charging Stations sends a TransactionEventRequest</p> <p>6. The OCTT responds with a TransactionEventResponse</p>
Tool validations	<ul style="list-style-type: none"> * Step 4: Message: GetTransactionStatusResponse - ongoingIndicator must be <i>true</i> - messagesInQueue must be <i>true</i> * Step 5: Message: TransactionEventRequest - eventType must be <i>Updated</i> - meterValues must be present. - offline must be <i>true</i> <p>Post scenario validations: N/a</p>	

Table 122. Test Case Id: TC_E_30_CS

Test case name	Check Transaction status - Transaction with id ongoing - without message in queue	
Test case Id	TC_E_30_CS	
Use case Id(s)	E14	
Requirement(s)	E14.FR.02,E14.FR.05	
System under test	Charging Station	
Description	The CSMS is able to request the status of a transaction and to find out whether there are queued transaction-related messages, using the GetTransactionStatusRequest message.	
Purpose	To verify if the Charging Station is able to correctly respond to a GetTransactionStatusRequest with a transactionId, while there is NO message queued belonging to an ongoing transaction with the requested id.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): State is <i>EnergyTransferStarted</i>	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with a GetTransactionStatusResponse	1. The OCTT sends a GetTransactionStatusRequest with transactionId <Generated transactionId from Before>
Tool validations	<ul style="list-style-type: none"> * Step 2: Message: GetTransactionStatusResponse - ongoingIndicator must be <i>true</i> - messagesInQueue must be <i>false</i> 	
	Post scenario validations: N/a	

Table 123. Test Case Id: TC_E_31_CS

Test case name	Check Transaction status - Transaction with id ended - with message in queue	
Test case Id	TC_E_31_CS	
Use case Id(s)	E14	
Requirement(s)	E14.FR.03,E14.FR.04	
System under test	Charging Station	
Description	The CSMS is able to request the status of a transaction and to find out whether there are queued transaction-related messages, using the GetTransactionStatusRequest message.	
Purpose	To verify if the Charging Station is able to correctly respond to a GetTransactionStatusRequest with a transactionId, while there is a message queued belonging to an ended transaction with the requested id.	
Prerequisite(s)	The Charging Station supports at least one authorization method described at the following Use cases; C01, C02, C04.	
Before (Preparations)	<p>Configuration State: OfflineThreshold is <Configured RetryBackOffWaitMinimum_duration> + 60.0 RetryBackOffWaitMinimum is <Configured RetryBackOffWaitMinimum_duration> RetryBackOffRandomRange is 0</p> <p><u>Note:</u> <i><Configured RetryBackOffWaitMinimum_duration> should be long enough to execute manual tasks after waiting for <Configured Transaction Duration> seconds</i></p> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>EnergyTransferStarted</i></p>	
Main (Test scenario)	Charging Station	CSMS
		The OCTT closes the WebSocket connection AND does not accept a reconnect.
	<u>Manual Action:</u> Present the same idToken as used to start the transaction.	
	<u>Manual Action:</u> Disconnect the EV and EVSE.	
	<u>Manual Action:</u> Drive EV out of parking bay. (Only needed if TxStopPoint is ParkingBayOccupancy)	
<u>Notes(s):</u> The tool will wait for <Configured Transaction Duration> seconds		The OCTT accepts reconnection attempt from the Charging Station.
	2. The Charging Station responds with a GetTransactionStatusResponse	1. The OCTT sends a GetTransactionStatusRequest with transactionId <Generated transactionId from Before>
		<u>Note:</u> <i>This step will be executed the moment the WebSocket connection is restored.</i>
	3. The Charging Stations sends a TransactionEventRequest	4. The OCTT responds with a TransactionEventResponse
<u>Note(s):</u> - The Charging Station will empty its Transaction message queue. This will contain all TransactionEventRequest messages from the Transaction.		

Test case name	Check Transaction status - Transaction with id ended - with message in queue
Tool validations	<ul style="list-style-type: none"> * Step 2: Message: GetTransactionStatusResponse - ongoingIndicator must be <i>false</i> - messagesInQueue must be <i>true</i> * Step 3: Message: TransactionEventRequest The tool validations from the reusable states need to be used to verify whether all required TransactionEventRequests have been received. From <i>StopAuthorized</i> through <i>ParkingBayUnoccupied</i> <p>Post scenario validations: N/a</p>

Table 124. Test Case Id: TC_E_32_CS

Test case name	Check Transaction status - Transaction with id ended - without message in queue	
Test case Id	TC_E_32_CS	
Use case Id(s)	E14	
Requirement(s)	E14.FR.03,E14.FR.05	
System under test	Charging Station	
Description	The CSMS is able to request the status of a transaction and to find out whether there are queued transaction-related messages, using the GetTransactionStatusRequest message.	
Purpose	To verify if the Charging Station is able to correctly respond to a GetTransactionStatusRequest with a transactionId, while there is NO message queued belonging to an ended transaction with the requested id.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): State is <i>EnergyTransferStarted, ParkingBayUnoccupied</i>	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with a GetTransactionStatusResponse	1. The OCTT sends a GetTransactionStatusRequest with transactionId <Generated transactionId from Before>
Tool validations	<ul style="list-style-type: none"> * Step 2: Message: GetTransactionStatusResponse - ongoingIndicator must be <i>false</i> - messagesInQueue must be <i>false</i> 	
	Post scenario validations: N/a	

Table 125. Test Case Id: TC_E_33_CS

Test case name	Check Transaction status - Without transactionId - with message in queue	
Test case Id	TC_E_33_CS	
Use case Id(s)	E14	
Requirement(s)	E14.FR.06,E14.FR.07	
System under test	Charging Station	
Description	The CSMS is able to request the status of a transaction and to find out whether there are queued transaction-related messages, using the GetTransactionStatusRequest message.	
Purpose	To verify if the Charging Station is able to correctly respond to a GetTransactionStatusRequest without a transactionId, while there is a message queued.	
Prerequisite(s)	N/a	
Before (Preparations)	<p>Configuration State:</p> <p>SampledDataTxUpdatedMeasurands is <Configured transaction_updated_metervalues_measurands></p> <p>SampledDataTxUpdatedInterval is <Configured transaction_updated_metervalues_interval></p> <p>OfflineThreshold is <Configured RetryBackOffWaitMinimum_duration> + 60.0</p> <p>RetryBackOffWaitMinimum is <Configured RetryBackOffWaitMinimum_duration></p> <p>RetryBackOffRandomRange is 0</p> <p>Note: <Configured RetryBackOffWaitMinimum_duration> must be greater than <Configured Transaction MeterValues interval></p> <p>Memory State:</p> <p>N/a</p> <p>Reusable State(s): State is <i>EnergyTransferStarted</i></p>	
Main (Test scenario)	<p>Charging Station</p> <p>4. The Charging Station responds with a GetTransactionStatusResponse</p> <p>Note(s): - The Charging Station will empty its Transaction message queue. This will contain one or more TransactionEventRequest messages</p>	<p>CSMS</p> <p>1. The OCTT closes the WebSocket connection AND does not accept a reconnect.</p> <p>2. The OCTT waits a number of seconds equal to <Configured RetryBackOffWaitMinimum_duration>, before accepting a reconnection attempt from the Charging Station.</p> <p>3. The OCTT sends a GetTransactionStatusRequest with transactionId omitted</p> <p>Note: <i>This step will be executed the moment the WebSocket connection is restored.</i></p> <p>5. The Charging Stations sends a TransactionEventRequest</p> <p>6. The OCTT responds with a TransactionEventResponse</p>
Tool validations	<ul style="list-style-type: none"> * Step 4: Message: GetTransactionStatusResponse - ongoingIndicator must be omitted - messagesInQueue must be <i>true</i> * Step 5: Message: TransactionEventRequest - eventType must be <i>Updated</i> - meterValues must be present. - offline must be <i>true</i> <p>Post scenario validations: N/a</p>	

Table 126. Test Case Id: TC_E_34_CS

Test case name	Check Transaction status - Without transactionId - without message in queue	
Test case Id	TC_E_34_CS	
Use case Id(s)	E14	
Requirement(s)	E14.FR.06,E14.FR.08	
System under test	Charging Station	
Description	The CSMS is able to request the status of a transaction and to find out whether there are queued transaction-related messages, using the GetTransactionStatusRequest message.	
Purpose	To verify if the Charging Station is able to correctly respond to a GetTransactionStatusRequest without a transactionId, while there is NO message queued.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with a GetTransactionStatusResponse	1. The OCTT sends a GetTransactionStatusRequest with transactionId omitted
Tool validations	<ul style="list-style-type: none"> * Step 2: Message: GetTransactionStatusResponse - ongoingIndicator must be omitted - messagesInQueue must be <i>false</i> 	
	Post scenario validations: N/a	

Table 127. Test Case Id: TC_E_40_CS

Test case name	Offline Behaviour - Connection loss during transaction	
Test case Id	TC_E_40_CS	
Use case Id(s)	E11	
Requirement(s)	E11.FR.01,E11.FR.02,E11.FR.06	
System under test	Charging Station	
Description	The Charging Station queues TransactionEvent messages to inform the CSMS that a transaction occurred while the Charging Station was Offline.	
Purpose	To verify if the Charging Station is able to queue TransactionEvent messages while it is offline.	
Prerequisite(s)	N/a	
Before (Preparations)	<p>Configuration State:</p> <p>SampledDataTxUpdatedMeasurands is <Configured transaction_updated_metervalues_measurands></p> <p>SampledDataTxUpdatedInterval is <Configured transaction_updated_metervalues_interval></p> <p>SampledDataEnabled is <i>true</i></p> <p>OfflineThreshold is <Configured RetryBackOffWaitMinimum_duration> + 60.0</p> <p>RetryBackOffWaitMinimum is <Configured RetryBackOffWaitMinimum_duration></p> <p>RetryBackOffRandomRange is 0</p> <p>Note: <Configured RetryBackOffWaitMinimum_duration> must be greater than <Configured Transaction MeterValues interval></p> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>EnergyTransferStarted</i></p>	
Main (Test scenario)	<p>Charging Station</p> <p>1. The OCTT closes the WebSocket connection AND does not accept a reconnect.</p> <p>2. The OCTT waits a number of seconds equal to <Configured RetryBackOffWaitMinimum_duration>, before accepting a reconnection attempt from the Charging Station.</p> <p>3. The Charging Stations sends a TransactionEventRequest</p> <p>Note(s): - The Charging Station will empty its Transaction message queue. This will contain one or more TransactionEventRequest messages</p>	<p>CSMS</p> <p>4. The OCTT responds with a TransactionEventResponse</p>
Tool validations	<ul style="list-style-type: none"> * Step 3: All messages: TransactionEventRequest - eventType must be <i>Updated</i> - meterValues must be present. - offline must be <i>true</i> <p>Post scenario validations: N/a</p>	

Table 128. Test Case Id: TC_E_41_CS

Test case name	Retry sending transaction message when failed - Max retry count reached	
Test case Id	TC_E_41_CS	
Use case Id(s)	E13	
Requirement(s)	E13.FR.01,E13.FR.02,E13.FR.03,E13.FR.04	
System under test	Charging Station	
Description	There are situations/issues why a CSMS might not accept a transaction related message, or does not reply within the MessageTimeout. Most are error scenarios. When something like this happens, the Charging Station SHALL retry the messages a couple of times.	
Purpose	To verify if the Charging Station is able to resend TransactionEvent messages until the max retry count is reached, when the CSMS does not reply.	
Prerequisite(s)	N/a	
Before (Preparations)	<p>Configuration State: MessageAttemptsTransactionEvent is <Configured message_attempts_transaction_event> (Must be > 1) MessageAttemptIntervalTransactionEvent is <Configured message_attempts_transaction_event_interval></p> <p>Memory State: N/a</p> <p>Reusable State(s): State is Authorized State is EVConnectedPreSession</p>	
Main (Test scenario)	<p>Charging Station</p> <p><u>Note(s): Step 1, 2, 3, & 4 are optional</u></p> <p>1. The Charging Stations sends a TransactionEventRequest with: - triggerReason SignedDataReceived</p> <p>3. The Charging Stations sends a TransactionEventRequest with: - triggerReason ChargingStateChanged - chargingState SuspendedEVSE</p> <p><u>Note(s): Step 5 is repeated for the configured number of times</u></p> <p>5. The Charging Stations sends a TransactionEventRequest</p>	<p>CSMS</p> <p>2. The OCTT responds with a TransactionEventResponse</p> <p>4. The OCTT responds with a TransactionEventResponse</p>
Tool validations	<ul style="list-style-type: none"> * Step 1: - triggerReason SignedDataReceived * Step 3: - triggerReason ChargingStateChanged - chargingState SuspendedEVSE * Step 5: - Needs to be send a number of times equal to <Configured message_attempts_transaction_event> with an interval of (<Configured message_attempts_transaction_event_interval> * the number of preceding transmissions of this same message) + <i>OCPPCommCtrlr.MessageTimeout.Default</i>. - The OCTT waits an additional MessageAttemptsTransactionEvent iteration where the interval is multiplied again, to validate if the Charging Station stops resending the TransactionRequest message(s). <p>Post scenario validations: N/a</p>	

Table 129. Test Case Id: TC_E_50_CS

Test case name	Retry sending transaction message when failed - Max retry count reached - CallError	
Test case Id	TC_E_50_CS	
Use case Id(s)	E13	
Requirement(s)	E13.FR.01,E13.FR.02,E13.FR.03,E13.FR.04	
System under test	Charging Station	
Description	There are situations/issues why a CSMS might not accept a transaction related message, or does not reply within the MessageTimeout. Most are error scenarios. When something like this happens, the Charging Station SHALL retry the messages a couple of times.	
Purpose	To verify if the Charging Station is able to resend TransactionEvent messages until the max retry count is reached, when the CSMS does not reply.	
Prerequisite(s)	N/a	
Before (Preparations)	<p>Configuration State: MessageAttemptsTransactionEvent is <Configured message_attempts_transaction_event> (Must be > 1) MessageAttemptIntervalTransactionEvent is <Configured message_attempts_transaction_event_interval></p> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>Authorized</i> State is <i>EVConnectedPreSession</i></p>	
Main (Test scenario)	<p>Charging Station</p> <p><u>Note(s): Step 1, 2, 3, & 4 are optional</u></p> <p>1. The Charging Stations sends a TransactionEventRequest with: - triggerReason <i>SignedDataReceived</i></p> <p>3. The Charging Stations sends a TransactionEventRequest with: - triggerReason <i>ChargingStateChanged</i> - chargingState <i>SuspendedEVSE</i></p> <p><u>Note(s): Step 5 is repeated for the configured number of times</u></p> <p>5. The Charging Stations sends a TransactionEventRequest</p>	<p>CSMS</p> <p>2. The OCTT responds with a TransactionEventResponse</p> <p>4. The OCTT responds with a TransactionEventResponse</p> <p>6. The OCTT responds with a CallError with errorCode <i>InternalError</i></p>
Tool validations	<ul style="list-style-type: none"> * Step 1: - triggerReason <i>SignedDataReceived</i> * Step 3: - triggerReason <i>ChargingStateChanged</i> - chargingState <i>SuspendedEVSE</i> * Step 5: - Needs to be send a number of times equal to <Configured message_attempts_transaction_event> with an interval of the <Configured message_attempts_transaction_event_interval> multiplied by the number of preceding transmissions of this same message. - The OCTT waits an additional MessageAttemptsTransactionEvent iteration where the interval is multiplied again, to validate if the Charging Station stops resending the TransactionRequest message(s). <p>Post scenario validations: N/a</p>	

Table 130. Test Case Id: TC_E_42_CS

Test case name	Retry sending transaction message when failed - Success before reaching the max retry count	
Test case Id	TC_E_42_CS	
Use case Id(s)	E13	
Requirement(s)	E13.FR.01,E13.FR.02,E13.FR.03	
System under test	Charging Station	
Description	There are situations/issues why a CSMS might not accept a transaction related message, or does not reply within the MessageTimeout. Most are error scenarios. When something like this happens, the Charging Station SHALL retry the messages a couple of times.	
Purpose	To verify if the Charging Station is able to resend TransactionEvent messages when the CSMS does not reply and stops resending after getting a response before the max retry count is reached.	
Prerequisite(s)	N/a	
Before (Preparations)	<p>Configuration State: MessageAttemptsTransactionEvent is <Configured message_attempts_transaction_event> (Must be > 2) MessageAttemptIntervalTransactionEvent is <Configured message_attempts_transaction_event_interval></p> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>Authorized</i> State is <i>EVConnectedPreSession</i></p>	
Main (Test scenario)	<p>Charging Station</p> <p><u>Note(s):</u> Step 1, 2, 3, & 4 are optional</p> <p>1. The Charging Stations sends a TransactionEventRequest with: - triggerReason <i>SignedDataReceived</i></p> <p>3. The Charging Stations sends a TransactionEventRequest with: - triggerReason <i>ChargingStateChanged</i> - chargingState <i>SuspendedEVSE</i></p> <p><u>Note(s):</u> The tool will ignore the first request and only respond to the second request</p> <p>5. The Charging Stations sends a TransactionEventRequest</p>	<p>CSMS</p> <p>2. The OCTT responds with a TransactionEventResponse</p> <p>4. The OCTT responds with a TransactionEventResponse</p> <p>6. The OCTT responds with a TransactionEventResponse</p>
Tool validations	<ul style="list-style-type: none"> * Step 1: - triggerReason <i>SignedDataReceived</i> * Step 3: - triggerReason <i>ChargingStateChanged</i> - chargingState <i>SuspendedEVSE</i> * Step 5: - Needs to be send a number of times equal to <Configured message_attempts_transaction_event> with an interval of (<Configured message_attempts_transaction_event_interval> * the number of preceding transmissions of this same message) + <i>OCPPCommCtrlr.MessageTimeout.Default</i>. - The OCTT waits an additional MessageAttemptsTransactionEvent iteration where the interval is multiplied again, to validate if the Charging Station stops resending the TransactionRequest message(s). <p>Post scenario validations: N/a</p>	

Table 131. Test Case Id: TC_E_51_CS

Test case name	Retry sending transaction message when failed - Success before reaching the max retry count - CallError									
Test case Id	TC_E_51_CS									
Use case Id(s)	E13									
Requirement(s)	E13.FR.01,E13.FR.02,E13.FR.03									
System under test	Charging Station									
Description	There are situations/issues why a CSMS might not accept a transaction related message, or does not reply within the MessageTimeout. Most are error scenarios. When something like this happens, the Charging Station SHALL retry the messages a couple of times.									
Purpose	To verify if the Charging Station is able to resend TransactionEvent messages when the CSMS does not reply and stops resending after getting a response before the max retry count is reached.									
Prerequisite(s)	N/a									
Before (Preparations)	<p>Configuration State: MessageAttemptsTransactionEvent is <Configured message_attempts_transaction_event> (Must be > 2) MessageAttemptIntervalTransactionEvent is <Configured message_attempts_transaction_event_interval></p> <p>Memory State: N/a</p> <p>Reusable State(s): State is Authorized State is EVConnectedPreSession</p>									
Main (Test scenario)	<p>Charging Station</p> <p><u>Note(s):</u> Step 1, 2, 3, & 4 are optional</p> <table border="1"> <tr> <td>1. The Charging Stations sends a TransactionEventRequest with: - triggerReason SignedDataReceived</td><td>2. The OCTT responds with a TransactionEventResponse</td></tr> <tr> <td>3. The Charging Stations sends a TransactionEventRequest with: - triggerReason ChargingStateChanged - chargingState SuspendedEVSE</td><td>4. The OCTT responds with a TransactionEventResponse</td></tr> <tr> <td colspan="2"><u>Note(s):</u> The tool will send a CallError with errorCode InternalError to all requests except for the second request, there a TransactionEventResponse is send</td></tr> <tr> <td>5. The Charging Stations sends a TransactionEventRequest</td><td>6. The OCTT responds with a TransactionEventResponse</td></tr> </table>	1. The Charging Stations sends a TransactionEventRequest with: - triggerReason SignedDataReceived	2. The OCTT responds with a TransactionEventResponse	3. The Charging Stations sends a TransactionEventRequest with: - triggerReason ChargingStateChanged - chargingState SuspendedEVSE	4. The OCTT responds with a TransactionEventResponse	<u>Note(s):</u> The tool will send a CallError with errorCode InternalError to all requests except for the second request, there a TransactionEventResponse is send		5. The Charging Stations sends a TransactionEventRequest	6. The OCTT responds with a TransactionEventResponse	CSMS
1. The Charging Stations sends a TransactionEventRequest with: - triggerReason SignedDataReceived	2. The OCTT responds with a TransactionEventResponse									
3. The Charging Stations sends a TransactionEventRequest with: - triggerReason ChargingStateChanged - chargingState SuspendedEVSE	4. The OCTT responds with a TransactionEventResponse									
<u>Note(s):</u> The tool will send a CallError with errorCode InternalError to all requests except for the second request, there a TransactionEventResponse is send										
5. The Charging Stations sends a TransactionEventRequest	6. The OCTT responds with a TransactionEventResponse									
Tool validations	<ul style="list-style-type: none"> * Step 1: - triggerReason SignedDataReceived * Step 3: - triggerReason ChargingStateChanged - chargingState SuspendedEVSE * Step 5: - Needs to be send a number of times equal to <Configured message_attempts_transaction_event> with an interval of <Configured message_attempts_transaction_event_interval> * the number of preceding transmissions of this same message. - The OCTT waits an additional MessageAttemptsTransactionEvent iteration where the interval is multiplied again, to validate if the Charging Station stops resending the TransactionRequest message(s). <p>Post scenario validations: N/a</p>									

Table 132. Test Case Id: TC_E_43_CS

Test case name	Offline Behaviour - Transaction during offline period	
Test case Id	TC_E_43_CS	
Use case Id(s)	E12	
Requirement(s)	E12.FR.01,E12.FR.02,E12.FR.06	
System under test	Charging Station	
Description	The Charging Station queues TransactionEvent messages to inform the CSMS that a transaction occurred while the Charging Station was Offline.	
Purpose	To verify if the Charging Station is able to queue TransactionEvent messages while it was offline.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	1. Execute Reusable State <i>TransactionEventsInQueueEnded</i>	
	2. The Charging Stations sends a TransactionEventRequest	3. The OCTT responds with a TransactionEventResponse
	<p>Note(s):</p> <p>- The Charging Station will empty its Transaction message queue. This will contain one or more TransactionEventRequest messages</p>	
Tool validations	<ul style="list-style-type: none"> * Step 2: All messages: TransactionEventRequest - offline must be <i>true</i> One of the messages: TransactionEventRequest - eventType Started One of the messages: TransactionEventRequest - eventType Ended 	
	<p>Post scenario validations: N/a</p>	

Table 133. Test Case Id: TC_E_44_CS

Test case name	Offline Behaviour - Stop transaction during offline period	
Test case Id	TC_E_44_CS	
Use case Id(s)	E08	
Requirement(s)	E08.FR.01,E08.FR.04,E08.FR.05,E08.FR.06,E08.FR.07,E08.FR.08	
System under test	Charging Station	
Description	The Charging Station queues TransactionEvent messages to inform the CSMS that a transaction occurred while the Charging Station was Offline.	
Purpose	To verify if the Charging Station is able to queue TransactionEvent messages when the transaction stopped while the Charging Station was offline.	
Prerequisite(s)	N/a	
Before (Preparations)	<p>Configuration State: OfflineThreshold is <Configured RetryBackOffWaitMinimum_duration> + 60.0 RetryBackOffWaitMinimum is <Configured RetryBackOffWaitMinimum_duration> RetryBackOffRandomRange is 0</p> <p><u>Note:</u> <i><Configured RetryBackOffWaitMinimum_duration should be long enough to execute manual tasks></i></p> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>EnergyTransferStarted</i></p>	
Main (Test scenario)	<p>Charging Station</p> <p>CSMS</p> <p>1. The OCTT closes the WebSocket connection AND does not accept a reconnect.</p> <p><u>Manual Action:</u> Present the same idToken as used to start the transaction.</p> <p><u>Manual Action:</u> Disconnect the EV and EVSE.</p> <p><u>Manual Action:</u> Drive EV out of parking bay. (Only needed if TxStopPoint is ParkingBayOccupancy)</p> <p>2. The OCTT accepts the reconnection attempt from the Charging Station.</p> <p>3. The Charging Stations sends a TransactionEventRequest</p> <p><u>Note(s):</u> - The Charging Station will empty its Transaction message queue. This will contain one or more TransactionEventRequest messages</p> <p>4. The OCTT responds with a TransactionEventResponse</p>	
Tool validations	<ul style="list-style-type: none"> * Step 3: All messages: TransactionEventRequest - offline must be <i>true</i> One of the messages: TransactionEventRequest - eventType <i>Ended</i> <p>Post scenario validations: N/a</p>	

Table 134. Test Case Id: TC_E_45_CS

Test case name	Offline Behaviour - Stop transaction during offline period - Same GroupId																	
Test case Id	TC_E_45_CS																	
Use case Id(s)	E08																	
Requirement(s)	E08.FR.02,E08.FR.04,E08.FR.05,E08.FR.06,E08.FR.07,E08.FR.08																	
System under test	Charging Station																	
Description	The Charging Station queues TransactionEvent messages to inform the CSMS that a transaction occurred while the Charging Station was Offline.																	
Purpose	To verify if the Charging Station is able to queue TransactionEvent messages when the transaction stopped by an idToken with the same groupIDToken, while the Charging Station was offline.																	
Prerequisite(s)	The Charging Station supports Authorization cache OR Local Authorization List.																	
Before (Preparations)	<p>Configuration State: OfflineThreshold is <Configured RetryBackOffWaitMinimum_duration> + 60.0 RetryBackOffWaitMinimum is <Configured RetryBackOffWaitMinimum_duration> RetryBackOffRandomRange is 0</p> <p>Note: <i><Configured RetryBackOffWaitMinimum_duration should be long enough to execute manual tasks></i></p> <p>Memory State: IdTokenCached for <Configured valid idtoken fields2> with <Configured GroupIdToken> IdTokenLocalAuthList for <Configured valid idtoken fields2> with <Configured GroupIdToken></p> <p>Reusable State(s): State is Authorized with <Configured GroupIdToken> Then proceed to state EnergyTransferStarted</p>																	
Main (Test scenario)	<table border="1"> <tr> <td>Charging Station</td> <td>CSMS</td> </tr> <tr> <td></td> <td>1. The OCTT closes the WebSocket connection AND does not accept a reconnect.</td> </tr> <tr> <td><u>Manual Action:</u> Present <Configured valid idtoken fields2>.</td> <td></td> </tr> <tr> <td><u>Manual Action:</u> Disconnect the EV and EVSE.</td> <td></td> </tr> <tr> <td><u>Manual Action:</u> Drive EV out of parking bay. (Only needed if TxStopPoint is ParkingBayOccupancy)</td> <td></td> </tr> <tr> <td></td> <td>2. The OCTT accepts the reconnection attempt from the Charging Station.</td> </tr> <tr> <td>3. The Charging Stations sends a TransactionEventRequest</td> <td>4. The OCTT responds with a TransactionEventResponse</td> </tr> <tr> <td>Note(s): - The Charging Station will empty its Transaction message queue. This will contain one or more TransactionEventRequest messages</td> <td></td> </tr> </table>	Charging Station	CSMS		1. The OCTT closes the WebSocket connection AND does not accept a reconnect.	<u>Manual Action:</u> Present <Configured valid idtoken fields2>.		<u>Manual Action:</u> Disconnect the EV and EVSE.		<u>Manual Action:</u> Drive EV out of parking bay. (Only needed if TxStopPoint is ParkingBayOccupancy)			2. The OCTT accepts the reconnection attempt from the Charging Station.	3. The Charging Stations sends a TransactionEventRequest	4. The OCTT responds with a TransactionEventResponse	Note(s): - The Charging Station will empty its Transaction message queue. This will contain one or more TransactionEventRequest messages		
Charging Station	CSMS																	
	1. The OCTT closes the WebSocket connection AND does not accept a reconnect.																	
<u>Manual Action:</u> Present <Configured valid idtoken fields2>.																		
<u>Manual Action:</u> Disconnect the EV and EVSE.																		
<u>Manual Action:</u> Drive EV out of parking bay. (Only needed if TxStopPoint is ParkingBayOccupancy)																		
	2. The OCTT accepts the reconnection attempt from the Charging Station.																	
3. The Charging Stations sends a TransactionEventRequest	4. The OCTT responds with a TransactionEventResponse																	
Note(s): - The Charging Station will empty its Transaction message queue. This will contain one or more TransactionEventRequest messages																		
Tool validations	<ul style="list-style-type: none"> * Step 3: All messages: TransactionEventRequest - offline must be <i>true</i> One of the messages: TransactionEventRequest - eventType <i>Ended</i> <p>Post scenario validations: N/a</p>																	

2.7. F Remote Control

Table 135. Test Case Id: TC_F_01_CS

Test case name	Remote start transaction - Cable plugin first	
Test case Id	TC_F_01_CS	
Use case Id(s)	F01	
Requirement(s)	F01.FR.03, F01.FR.04, F01.FR.05, F01.FR.13, F01.FR.17, F01.FR.19, F02.FR.01	
System under test	Charging Station	
Description	OCPP 2.x.x allows an EV driver to either first connect the EV and EVSE OR wait for/trigger a RequestStartTransactionRequest. Both sequences will result in being able to charge.	
Purpose	To verify if the Charging Station is able to start a charging session when the EV driver first connects the EV and EVSE, before receiving a RequestStartTransactionRequest message.	
Prerequisite(s)	- The Charging Station does NOT have a cable lock, which prevents the EV driver to connect the EV and EVSE before authorization.	
Before (Preparations)	<p>Configuration State: AuthCtrlIr.AuthEnabled is <i>true</i> (If implemented AND ReadWrite) AuthCtrlIr.DisableRemoteAuthorization is <i>false</i> (If implemented)</p> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>EVConnectedPreSession</i></p>	
Main (Test scenario)	Charging Station	CSMS
	1. Execute Reusable State <i>Authorized</i> (remote) 2. Execute Reusable State <i>EnergyTransferStarted</i>	
Tool validations	N/a Post scenario validations: N/a	

Table 136. Test Case Id: TC_F_02_CS

Test case name	Remote start transaction - Remote start first - AuthorizeRemoteStart is true	
Test case Id	TC_F_02_CS	
Use case Id(s)	F02	
Requirement(s)	F02.FR.01, F01.FR.01	
System under test	Charging Station	
Description	OCPP 2.x.x allows an EV driver to either first wait for/trigger a RequestStartTransactionRequest OR connect the EV and EVSE. Both sequences will result in being able to charge.	
Purpose	To verify if the Charging Station is able to start a charging session when the Charging Stations receives a RequestStartTransactionRequest message (while AuthorizeRemoteStart is true), before the EV driver connects the EV and EVSE (within the connectionTimeout). The Charging Station has to authorize beforehand like a local action to start a transaction.	
Prerequisite(s)	<ul style="list-style-type: none"> - AuthEnabled is NOT implemented with mutability ReadOnly and the value set to false AND - AuthorizeRemoteStart is NOT implemented with mutability ReadOnly and the value set to false 	
Before (Preparations)	<p>Configuration State: AuthCtrlr.AuthEnabled is <i>true</i> (If implemented AND ReadWrite) AuthCtrlr.DisableRemoteAuthorization is <i>false</i> (If implemented) AuthorizeRemoteStart is <i>true</i> (If ReadWrite)</p> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>ParkingBayOccupied</i> (Optional state)</p>	
Main (Test scenario)	Charging Station	CSMS
	1. Execute Reusable State <i>Authorized</i> (remote) 2. Execute Reusable State <i>EnergyTransferStarted</i>	
Tool validations	N/a Post scenario validations: N/a	

Table 137. Test Case Id: TC_F_03_CS

Test case name	Remote start transaction - Remote start first - AuthorizeRemoteStart is false	
Test case Id	TC_F_03_CS	
Use case Id(s)	F02	
Requirement(s)	F02.FR.01, F01.FR.02	
System under test	Charging Station	
Description	OCPP 2.x.x allows an EV driver to either first wait for/trigger a RequestStartTransactionRequest OR connect the EV and EVSE. Both sequences will result in being able to charge.	
Purpose	To verify if the Charging Station is able to start a charging session when the Charging Stations receives a RequestStartTransactionRequest message (while AuthorizeRemoteStart is false), before the EV driver connects the EV and EVSE (within the connectionTimeout). The Charging station does NOT have to authorize beforehand like a local action to start a transaction.	
Prerequisite(s)	AuthorizeRemoteStart is NOT implemented with mutability ReadOnly and the value set to true	
Before (Preparations)	<p>Configuration State: AuthCtrlr.AuthEnabled is <i>true</i> (If implemented AND ReadWrite) AuthCtrlr.DisableRemoteAuthorization is <i>false</i> (If implemented) AuthorizeRemoteStart is <i>false</i> (If ReadWrite)</p> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>ParkingBayOccupied</i> (Optional state)</p>	
Main (Test scenario)	Charging Station	CSMS
	1. Execute Reusable State <i>Authorized</i> (remote) 2. Execute Reusable State <i>EnergyTransferStarted</i>	
Tool validations	N/a Post scenario validations: N/a	

Table 138. Test Case Id: TC_F_04_CS

Test case name	Remote start transaction - Remote start first - Cable plugin timeout	
Test case Id	TC_F_04_CS	
Use case Id(s)	F02, E03	
Requirement(s)	F02.FR.01, E03.FR.01, E03.FR.05	
System under test	Charging Station	
Description	OCPP 2.x.x allows an EV driver to either first wait for/trigger a RequestStartTransactionRequest OR connect the EV and EVSE. Both sequences will result in being able to charge.	
Purpose	To verify if the Charging Station is able to deauthorize the transaction after the EVConnectionTimeout has been reached.	
Prerequisite(s)	N/a	
Before (Preparations)	<p>Configuration State:</p> <ul style="list-style-type: none"> - TxCtrlr.EVConnectionTimeOut is <i><Configured ev_connection_timeout></i> - AuthCtrlr.AuthEnabled is <i>true</i> (If implemented AND ReadWrite) AuthCtrlr.DisableRemoteAuthorization is <i>false</i> (If implemented) - TxCtrlr.TxStartPoint is <i>ParkingBayOccupancy</i> OR <i>Authorized</i> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>Authorized</i> (remote)</p>	
Main (Test scenario)	<p>Charging Station</p> <p>1. The Charging Station sends a TransactionEventRequest</p> <p>Note(s):</p> <ul style="list-style-type: none"> - This step needs to be executed after the <i><Configured ev_connection_timeout></i> expires, if the transaction has been started. So in the case TxStartPoint contains <i>ParkingBayOccupancy</i> OR <i>Authorized</i> <p>Note(s): Optionally the Charging Station can send a StatusNotificationRequest or NotifyEventRequest with status Available</p> <p>3. Execute Reusable State <i>Authorized</i> (remote)</p> <p>Note(s):</p> <ul style="list-style-type: none"> - This step is executed to verify if the EVSE is actually ready to start another charging session. 	<p>CSMS</p> <p>2. The OCTT responds with a TransactionEventResponse</p>
Tool validations	<ul style="list-style-type: none"> * Step 1: Message: TransactionEventRequest - triggerReason must be <i>EVConnectTimeout</i> - eventType must be <i>Ended</i> AND <p>Post scenario validations: N/a</p>	

Table 139. Test Case Id: TC_F_05_CS

Test case name	Remote unlock Connector - With ongoing transaction	
Test case Id	TC_F_05_CS	
Use case Id(s)	F05	
Requirement(s)	F05.FR.01, F05.FR.02	
System under test	Charging Station	
Description	This test case describes how the CSMS can be requested to sent an UnlockConnectorRequest to the charging station. It sometimes happens that a connector of a Charging Station socket does not unlock correctly. This happens most of the time when there is tension on the charging cable. This means the driver cannot unplug his charging cable from the Charging Station. To help a driver, the CSO can send a UnlockConnectorRequest to the Charging Station. The Charging Station will then try to unlock the connector again.	
Purpose	To verify if the Chargin Station is able to ignore the UnlockConnectorRequest whith an ongoing transaction as described at the OCPP specification.	
Prerequisite(s)	The Charging Station has a connector lock.	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Charging State: Transaction is ongoing on <Configured Connector> State is EnergyTransferStarted	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with a UnlockConnectorResponse	1. The OCTT sends a UnlockConnectorRequest with evseld <Configured evseld> connectorId <Configured connectorId>
Tool validations	* Step 2: Message UnlockConnectorResponse - status OngoingAuthorizedTransaction	
	Post scenario validations: - N/a	

Table 140. Test Case Id: TC_F_06_CS

Test case name	Remote unlock Connector - Without ongoing transaction - Accepted	
Test case Id	TC_F_06_CS	
Use case Id(s)	F05	
Requirement(s)	F05.FR.01, F05.FR.04	
System under test	Charging Station	
Description	This test case describes how the CSMS can be requested to sent an UnlockConnectorRequest to the charging station. It sometimes happens that a connector of a Charging Station socket does not unlock correctly. This happens most of the time when there is tension on the charging cable. This means the driver cannot unplug his charging cable from the Charging Station. To help a driver, the CSO can send a UnlockConnectorRequest to the Charging Station. The Charging Station will then try to unlock the connector again.	
Purpose	To verify if the Charging Station is able to successfully unlock a connector without ongoing transaction as described in the OCPP specification.	
Prerequisite(s)	The Charging Station has a connector lock.	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with a UnlockConnectorResponse	1. The OCTT sends a UnlockConnectorRequest with evsId <Configured evsId> connectorId <Configured connectorId>
Tool validations	* Step 2: Message UnlockConnectorResponse - status Unlocked	
	Post scenario validations: - N/a	

Table 141. Test Case Id: TC_F_07_CS

Test case name	Remote unlock Connector - Without ongoing transaction - No cable connected	
Test case Id	TC_F_07_CS	
Use case Id(s)	F05	
Requirement(s)	F05.FR.01, F05.FR.06	
System under test	Charging Station	
Description	This test case describes how the CSMS can be requested to sent an UnlockConnectorRequest to the charging station. It sometimes happens that a connector of a Charging Station socket does not unlock correctly. This happens most of the time when there is tension on the charging cable. This means the driver cannot unplug his charging cable from the Charging Station. To help a driver, the CSO can send a UnlockConnectorRequest to the Charging Station. The Charging Station will then try to unlock the connector again.	
Purpose	To verify if the Chargin Station is able to perform the remote unlock connector mechanism and report the result without ongoing transaction while no cable is connected as described at the OCPP specification.	
Prerequisite(s)	The Charging Station has a connector lock.	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Charging State: No cable connected at <Configured Connector>	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with a UnlockConnectorResponse	1. The OCTT sends a UnlockConnectorRequest with evsId <Configured evsId> connectorId <Configured connectorId>
Tool validations	<ul style="list-style-type: none"> * Step 2: Message UnlockConnectorResponse - status Unlocked 	
	Post scenario validations: <ul style="list-style-type: none"> - N/a 	

Table 142. Test Case Id: TC_F_08_CS

Test case name	Remote stop transaction - Success	
Test case Id	TC_F_08_CS	
Use case Id(s)	F03	
Requirement(s)	F03.FR.02, F03.FR.03, F03.FR.07, F03.FR.09	
System under test	Charging Station	
Description	The CSMS is able to stop a charging session remotely by sending a RequestStopTransactionRequest to the Charging Station.	
Purpose	To verify if the Charging Station is able to stop a charging session when it receives a RequestStopTransactionRequest message.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): State is <i>EnergyTransferStarted</i>	
Main (Test scenario)	Charging Station	CSMS
	1. Execute Reusable State <i>StopAuthorized</i> (remote)	
Tool validations	N/a	
	Post scenario validations: N/a	

Table 143. Test Case Id: TC_F_09_CS

Test case name	Remote stop transaction - Rejected	
Test case Id	TC_F_09_CS	
Use case Id(s)	F03	
Requirement(s)	F03.FR.08	
System under test	Charging Station	
Description	The CSMS is able to stop a charging session remotely by sending a RequestStopTransactionRequest to the Charging Station.	
Purpose	To verify if the Charging Station will reject a RequestStopTransactionRequest message, if it contains a transactionId that cannot be matched to an active transaction.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): State is <i>EnergyTransferStarted</i>	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with a RequestStopTransactionResponse	1. The OCTT sends a RequestStopTransactionRequest with transactionId <Different transactionId than provided by the Charging Station in TransactionEventRequest>
Tool validations	<ul style="list-style-type: none"> * Step 2: <p>Message: RequestStopTransactionResponse - status must be <i>Rejected</i></p>	
	Post scenario validations: N/a	

Table 144. Test Case Id: TC_F_10_CS

Test case name	Remote unlock Connector - Without ongoing transaction - UnknownConnector	
Test case Id	TC_F_10_CS	
Use case Id(s)	F05	
Requirement(s)	F05.FR.03	
System under test	Charging Station	
Description	This test case describes how the CSMS can be requested to sent an UnlockConnectorRequest to the charging station. It sometimes happens that a connector of a Charging Station socket does not unlock correctly. This happens most of the time when there is tension on the charging cable. This means the driver cannot unplug his charging cable from the Charging Station. To help a driver, the CSO can send a UnlockConnectorRequest to the Charging Station. The Charging Station will then try to unlock the connector again.	
Purpose	To verify if the Charging Station is able to respond with a UnlockConnectorRequest with status <i>UnknownConnector</i> when the requested connector is unknown as described in the OCPP specification.	
Prerequisite(s)	The Charging Station has a connector lock.	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with a UnlockConnectorResponse	1. The OCTT sends a UnlockConnectorRequest with evselId <Configured evselId> connectorId 999
Tool validations	<ul style="list-style-type: none"> * Step 2: Message UnlockConnectorResponse - status UnknownConnector 	
	Post scenario validations: <ul style="list-style-type: none"> - N/a 	

Table 145. Test Case Id: TC_F_11_CS

Test case name	Trigger message - MeterValues - Specific EVSE	
Test case Id	TC_F_11_CS	
Use case Id(s)	F06	
Requirement(s)	F06.FR.02,F06.FR.04,F06.FR.05,F06.FR.06,F06.FR.10	
System under test	Charging Station	
Description	The CSMS can request a Charging Station to send Charging Station-initiated messages. In the request the CSMS indicates which message it wishes to receive.	
Purpose	To verify if the Charging Station is able to send a MeterValuesRequest message for a specific EVSE, after receiving a TriggerMessageRequest message.	
Prerequisite(s)	The Charging Station supports sending MeterValues triggered by a TriggerMessageRequest.	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): N/a	
Main (Test scenario)	Charging Station 2. The Charging Station responds with a TriggerMessageResponse 3. The Charging Station sends a MeterValuesRequest	CSMS 1. The OCTT sends a TriggerMessageRequest With requestedMessage MeterValues evse.id <Configured evseld> 4. The OCTT responds with a MeterValuesResponse
Tool validations	* Step 2: Message: TriggerMessageResponse - status must be Accepted * Step 3: Message: MeterValuesRequest - evseld must be <Configured evseld> - meterValue[0].sampledValue[0].context must be <i>Trigger</i> Post scenario validations: N/a	

Table 146. Test Case Id: TC_F_12_CS

Test case name	Trigger message - MeterValues - All EVSE	
Test case Id	TC_F_12_CS	
Use case Id(s)	F06	
Requirement(s)	F06.FR.04,F06.FR.05,F06.FR.06,F06.FR.10,F06.FR.11	
System under test	Charging Station	
Description	The CSMS can request a Charging Station to send Charging Station-initiated messages. In the request the CSMS indicates which message it wishes to receive.	
Purpose	To verify if the Charging Station is able to send a MeterValuesRequest message for all EVSE, after receiving a TriggerMessageRequest message.	
Prerequisite(s)	The Charging Station supports sending MeterValues triggered by a TriggerMessageRequest.	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): N/a	
Main (Test scenario)	Charging Station 2. The Charging Station responds with a TriggerMessageResponse 3. The Charging Station sends a MeterValuesRequest <u>Note(s):</u> - This step needs to be executed for every EVSE.	CSMS 1. The OCTT sends a TriggerMessageRequest With requestedMessage MeterValues evse is omitted 4. The OCTT responds with a MeterValuesResponse
Tool validations	* Step 2: Message: TriggerMessageResponse - status must be Accepted * Step 3: Message: MeterValuesRequest - meterValue[0].sampledValue[0].context must be <i>Trigger</i> Post scenario validations: N/a	

Table 147. Test Case Id: TC_F_13_CS

Test case name	Trigger message - TransactionEvent - Specific EVSE	
Test case Id	TC_F_13_CS	
Use case Id(s)	F06	
Requirement(s)	F06.FR.02,F06.FR.04,F06.FR.05,F06.FR.07,F06.FR.10	
System under test	Charging Station	
Description	The CSMS can request a Charging Station to send Charging Station-initiated messages. In the request the CSMS indicates which message it wishes to receive.	
Purpose	To verify if the Charging Station is able to send a TransactionEventRequest message for a specific EVSE, after receiving a TriggerMessageRequest message.	
Prerequisite(s)	The Charging Station supports sending TransactionEvents triggered by a TriggerMessageRequest.	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): State is <i>EnergyTransferStarted</i>	
Main (Test scenario)	Charging Station 2. The Charging Station responds with a TriggerMessageResponse 3. The Charging Station sends a TransactionEventRequest	CSMS 1. The OCTT sends a TriggerMessageRequest With requestedMessage <i>TransactionEvent evse.id <Configured evseld></i> 4. The OCTT responds with a TransactionEventResponse
Tool validations	<ul style="list-style-type: none"> * Step 2: Message: TriggerMessageResponse - status must be <i>Accepted</i> * Step 3: Message: TransactionEventRequest - evse.id must be omitted or <i><Configured evseld></i> - triggerReason must be <i>Trigger</i> - transactionInfo.chargingState must be <i>Charging</i> - meterValue must be present - meterValue[0].sampledValue[0].context must be <i>Trigger</i> <p>Post scenario validations: N/a</p>	

Table 148. Test Case Id: TC_F_14_CS

Test case name	Trigger message - TransactionEvent - All EVSE	
Test case Id	TC_F_14_CS	
Use case Id(s)	F06	
Requirement(s)	F06.FR.04,F06.FR.05,F06.FR.07,F06.FR.10,F06.FR.11	
System under test	Charging Station	
Description	The CSMS can request a Charging Station to send Charging Station-initiated messages. In the request the CSMS indicates which message it wishes to receive.	
Purpose	To verify if the Charging Station is able to send a TransactionEventRequest message for all EVSE, after receiving a TriggerMessageRequest message.	
Prerequisite(s)	The Charging Station supports sending TransactionEvents triggered by a TriggerMessageRequest.	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): State is <i>EnergyTransferStarted</i> for all EVSE	
Main (Test scenario)	Charging Station 2. The Charging Station responds with a TriggerMessageResponse 3. The Charging Station sends a TransactionEventRequest <u>Note(s):</u> - This step needs to be executed for every EVSE.	CSMS 1. The OCTT sends a TriggerMessageRequest With requestedMessage <i>TransactionEvent</i> evse is omitted 4. The OCTT responds with a TransactionEventResponse
Tool validations	* Step 2: Message: TriggerMessageResponse - status must be Accepted * Step 3: Message: TransactionEventRequest - evse.id must be <Configured evselid> - triggerReason must be <i>Trigger</i> - transactionInfo.chargingState must be <i>Charging</i> - meterValue must be present - meterValue[0].sampledValue[0].context must be <i>Trigger</i> Post scenario validations: N/a	

Table 149. Test Case Id: TC_F_15_CS

Test case name	Trigger message - LogStatusNotification - Idle	
Test case Id	TC_F_15_CS	
Use case Id(s)	F06	
Requirement(s)	F06.FR.04,F06.FR.05,F06.FR.10,F06.FR.15	
System under test	Charging Station	
Description	The CSMS can request a Charging Station to send Charging Station-initiated messages. In the request the CSMS indicates which message it wishes to receive.	
Purpose	To verify if the Charging Station is able to send a LogStatusNotificationRequest with status Idle, after receiving a TriggerMessageRequest while NOT uploading a log file.	
Prerequisite(s)	The Charging Station supports sending LogStatusNotifications triggered by a TriggerMessageRequest.	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with a TriggerMessageResponse	1. The OCTT sends a TriggerMessageRequest With requestedMessage LogStatusNotification
	3. The Charging Station sends a LogStatusNotificationRequest	4. The OCTT responds with a LogStatusNotificationResponse
Tool validations	<ul style="list-style-type: none"> * Step 2: Message: TriggerMessageResponse - status must be <i>Accepted</i> * Step 3: Message: LogStatusNotificationRequest - status must be <i>Idle</i> 	
	Post scenario validations: N/a	

Table 150. Test Case Id: TC_F_16_CS

Test case name	Trigger message - LogStatusNotification - Uploading	
Test case Id	TC_F_16_CS	
Use case Id(s)	F06	
Requirement(s)	F06.FR.04,F06.FR.05,F06.FR.10,F06.FR.14	
System under test	Charging Station	
Description	The CSMS can request a Charging Station to send Charging Station-initiated messages. In the request the CSMS indicates which message it wishes to receive.	
Purpose	To verify if the Charging Station is able to send a LogStatusNotificationRequest with status Uploading, after receiving a TriggerMessageRequest while uploading a log file.	
Prerequisite(s)	The Charging Station supports sending LogStatusNotifications triggered by a TriggerMessageRequest.	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with a GetLogResponse	1. The OCTT sends a GetLogRequest With logType DiagnosticsLog log.remoteLocation is <Configured log_location>
	3. The Charging Station sends a LogStatusNotificationRequest	4. The OCTT responds with a LogStatusNotificationResponse
	6. The Charging Station responds with a TriggerMessageResponse	5. The OCTT sends a TriggerMessageRequest With requestedMessage LogStatusNotification
	7. The Charging Station sends a LogStatusNotificationRequest	8. The OCTT responds with a LogStatusNotificationResponse
Tool validations	<ul style="list-style-type: none"> * Step 2: Message: GetLogResponse - status must be <i>Accepted</i> * Step 3: Message: LogStatusNotificationRequest - status must be <i>Uploading</i> * Step 6: Message: TriggerMessageResponse - status must be <i>Accepted</i> * Step 7: Message: LogStatusNotificationRequest - status must be <i>Uploading</i> <p>Post scenario validations: N/a</p>	

Table 151. Test Case Id: TC_F_17_CS

Test case name	Trigger message - FirmwareStatusNotification - Specific EVSE not relevant	
Test case Id	TC_F_17_CS	
Use case Id(s)	F06	
Requirement(s)	F06.FR.03,F06.FR.04,F06.FR.05,F06.FR.10	
System under test	Charging Station	
Description	The CSMS can request a Charging Station to send Charging Station-initiated messages. In the request the CSMS indicates which message it wishes to receive.	
Purpose	To verify if the Charging Station is able to send a FirmwareStatusNotificationRequest, after receiving a TriggerMessageRequest even when the CSMS an evseld which is not relevant for the requestedMessage FirmwareStatusNotification.	
Prerequisite(s)	The Charging Station supports sending FirmwareStatusNotifications triggered by a TriggerMessageRequest.	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with a TriggerMessageResponse	1. The OCTT sends a TriggerMessageRequest With requestedMessage FirmwareStatusNotification evse.id is <Configured evseld>
	3. The Charging Station sends a FirmwareStatusNotificationRequest	4. The OCTT responds with a FirmwareStatusNotificationResponse
Tool validations	* Step 2: Message: TriggerMessageResponse - status must be <i>Accepted</i> * Step 3: Message: FirmwareStatusNotificationRequest - status must be <i>Idle</i> Post scenario validations: N/a	

Table 152. Test Case Id: TC_F_18_CS

Test case name	Trigger message - FirmwareStatusNotification - Idle	
Test case Id	TC_F_18_CS	
Use case Id(s)	F06	
Requirement(s)	F06.FR.04,F06.FR.05,F06.FR.10,F06.FR.16,L01.FR.25	
System under test	Charging Station	
Description	The CSMS can request a Charging Station to send Charging Station-initiated messages. In the request the CSMS indicates which message it wishes to receive.	
Purpose	To verify if the Charging Station is able to send a FirmwareStatusNotificationRequest with status Idle, after receiving a TriggerMessageRequest while NOT downloading a firmware file.	
Prerequisite(s)	The Charging Station supports sending FirmwareStatusNotifications triggered by a TriggerMessageRequest.	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with a TriggerMessageResponse	1. The OCTT sends a TriggerMessageRequest With requestedMessage FirmwareStatusNotification
	3. The Charging Station sends a FirmwareStatusNotificationRequest	4. The OCTT responds with a FirmwareStatusNotificationResponse
	* Step 2: Message: TriggerMessageResponse - status must be <i>Accepted</i> * Step 3: Message: FirmwareStatusNotificationRequest - status must be <i>Idle</i>	
Tool validations	Post scenario validations: N/a	

Table 153. Test Case Id: TC_F_19_CS

Test case name	Trigger message - FirmwareStatusNotification - Downloading	
Test case Id	TC_F_19_CS	
Use case Id(s)	F06	
Requirement(s)	F06.FR.04,F06.FR.05,F06.FR.10,L01.FR.26	
System under test	Charging Station	
Description	The CSMS can request a Charging Station to send Charging Station-initiated messages. In the request the CSMS indicates which message it wishes to receive.	
Purpose	To verify if the Charging Station is able to send a FirmwareStatusNotificationRequest with status Downloading, after receiving a TriggerMessageRequest while downloading a firmware file.	
Prerequisite(s)	The Charging Station supports sending FirmwareStatusNotifications triggered by a TriggerMessageRequest.	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with a UpdateFirmwareResponse	1. The OCTT sends a UpdateFirmwareRequest <i>firmware.location</i> is <Configured firmware_location> <i>firmware.retrieveDateTime</i> is <Current dateTime - 2 hours> <i>firmware.installDateTime</i> is omitted <i>firmware.signingCertificate</i> is <Configured signingCertificate> <i>firmware.signature</i> is <Configured signature>
	3. The Charging Station sends a FirmwareStatusNotificationRequest	4. The OCTT responds with a FirmwareStatusNotificationResponse
	6. The Charging Station responds with a TriggerMessageResponse	5. The OCTT sends a TriggerMessageRequest With requestedMessage <i>FirmwareStatusNotification</i>
	7. The Charging Station sends a FirmwareStatusNotificationRequest	8. The OCTT responds with a FirmwareStatusNotificationResponse
Tool validations	* Step 2: Message: UpdateFirmwareResponse - status must be Accepted * Step 3: Message: FirmwareStatusNotificationRequest - status must be Downloading * Step 6: Message: TriggerMessageResponse - status must be Accepted * Step 7: Message: FirmwareStatusNotificationRequest - status must be Downloading Post scenario validations: N/a	

Table 154. Test Case Id: TC_F_20_CS

Test case name	Trigger message - Heartbeat	
Test case Id	TC_F_20_CS	
Use case Id(s)	F06	
Requirement(s)	F06.FR.04,F06.FR.05,F06.FR.10	
System under test	Charging Station	
Description	The CSMS can request a Charging Station to send Charging Station-initiated messages. In the request the CSMS indicates which message it wishes to receive.	
Purpose	To verify if the Charging Station is able to send a HeartbeatRequest, after receiving a TriggerMessageRequest.	
Prerequisite(s)	The Charging Station supports sending Heartbeats triggered by a TriggerMessageRequest.	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): N/a	
Main (Test scenario)	Charging Station 2. The Charging Station responds with a TriggerMessageResponse 3. The Charging Station sends a HeartbeatRequest	CSMS 1. The OCTT sends a TriggerMessageRequest With requestedMessage Heartbeat 4. The OCTT responds with a HeartbeatResponse
Tool validations	* Step 2: Message: TriggerMessageResponse - status must be Accepted Post scenario validations: N/a	

Table 155. Test Case Id: TC_F_23_CS

Test case name	Trigger message - StatusNotification - Specific EVSE - Available	
Test case Id	TC_F_23_CS	
Use case Id(s)	F06	
Requirement(s)	F06.FR.02,F06.FR.04,F06.FR.05,F06.FR.10	
System under test	Charging Station	
Description	The CSMS can request a Charging Station to send Charging Station-initiated messages. In the request the CSMS indicates which message it wishes to receive.	
Purpose	To verify if the Charging Station is able to send a StatusNotificationRequest message for a specific available EVSE/Connector, after receiving a TriggerMessageRequest message.	
Prerequisite(s)	The Charging Station supports sending StatusNotifications triggered by a TriggerMessageRequest.	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): N/a	
Main (Test scenario)	Charging Station 2. The Charging Station responds with a TriggerMessageResponse	CSMS 1. The OCTT sends a TriggerMessageRequest With requestedMessage StatusNotification evse.id <Configured evselId> evse.connectorId <Configured connectorId>
	3. The Charging Station notifies the CSMS about the current state of the connector.	4. The OCTT responds accordingly.
Tool validations	* Step 2: Message: TriggerMessageResponse - status must be Accepted * Step 3: Message: StatusNotificationRequest - connectorStatus Available Message: NotifyEventRequest - eventData[0].trigger Delta - eventData[0].actualValue "Available" - eventData[0].component.name "Connector" - eventData[0].variable.name "AvailabilityState" Post scenario validations: N/a	

Table 156. Test Case Id: TC_F_24_CS

Test case name	Trigger message - StatusNotification - Specific EVSE - Occupied	
Test case Id	TC_F_24_CS	
Use case Id(s)	F06	
Requirement(s)	F06.FR.02,F06.FR.04,F06.FR.05,F06.FR.10	
System under test	Charging Station	
Description	The CSMS can request a Charging Station to send Charging Station-initiated messages. In the request the CSMS indicates which message it wishes to receive.	
Purpose	To verify if the Charging Station is able to send a StatusNotificationRequest message for a specific occupied EVSE/Connector, after receiving a TriggerMessageRequest message.	
Prerequisite(s)	The Charging Station supports sending StatusNotifications triggered by a TriggerMessageRequest.	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): State is <i>EVConnectedPreSession</i>	
Main (Test scenario)	Charging Station 2. The Charging Station responds with a TriggerMessageResponse 3. The Charging Station notifies the CSMS about the current state of the connector.	CSMS 1. The OCTT sends a TriggerMessageRequest With requestedMessage <i>StatusNotification evse.id <Configured evseld> evse.connectorId <Configured connectorId></i> 4. The OCTT responds accordingly.
Tool validations	* Step 2: Message: TriggerMessageResponse - status must be <i>Accepted</i> * Step 3: Message: StatusNotificationRequest - connectorStatus <i>Occupied</i> Message: NotifyEventRequest - eventData[0].trigger <i>Delta</i> - eventData[0].actualValue " <i>Occupied</i> " - eventData[0].component.name " <i>Connector</i> " - eventData[0].variable.name " <i>AvailabilityState</i> " Post scenario validations: N/a	

Table 157. Test Case Id: TC_F_26_CS

Test case name	Trigger message - BootNotification - Rejected	
Test case Id	TC_F_26_CS	
Use case Id(s)	F06	
Requirement(s)	F06.FR.04,F06.FR.05,F06.FR.17	
System under test	Charging Station	
Description	The CSMS can request a Charging Station to send Charging Station-initiated messages. In the request the CSMS indicates which message it wishes to receive.	
Purpose	To verify if the Charging Station rejects resending a BootNotificationRequest, when it has already received an accepted on a previously sent BootNotification, after receiving a TriggerMessageRequest.	
Prerequisite(s)	The Charging Station supports sending BootNotification triggered by a TriggerMessageRequest.	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): N/a	
Main (Test scenario)	Charging Station 2. The Charging Station responds with a TriggerMessageResponse	CSMS 1. The OCTT sends a TriggerMessageRequest With requestedMessage <i>BootNotification</i>
Tool validations	* Step 2: Message: TriggerMessageResponse - status must be <i>Rejected</i> Post scenario validations: N/a	

Table 158. Test Case Id: TC_F_27_CS

Test case name	Trigger message - NotImplemented	
Test case Id	TC_F_27_CS	
Use case Id(s)	F06	
Requirement(s)	F06.FR.08	
System under test	Charging Station	
Description	The CSMS can request a Charging Station to send Charging Station-initiated messages. In the request the CSMS indicates which message it wishes to receive.	
Purpose	To verify if the Charging Station is able to report it has not implemented sending a SignCombinedCertificateRequest, after receiving a TriggerMessageRequest.	
Prerequisite(s)	The Charging Station does NOT support sending SignCombinedCertificates triggered by a TriggerMessageRequest.	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with a TriggerMessageResponse	1. The OCTT sends a TriggerMessageRequest With requestedMessage <i>SignCombinedCertificate</i>
Tool validations	* Step 2: Message: TriggerMessageResponse - status must be <i>NotImplemented</i>	
	Post scenario validations: N/a	

2.8. G Availability

Table 159. Test Case Id: TC_G_01_CS

Test case name	Connector status Notification - Available to Occupied	
Test case Id	TC_G_01_CS	
Use case Id(s)	G01, N07	
Requirement(s)	G01.FR.01, N07.FR.19	
System under test	Charging Station	
Description	A Charging Station sends a notification to the CSMS to inform the CSMS about a Connector status change. This can be done in two ways. Via a StatusNotificationRequest or a NotifyEventRequest from the device model.	
Purpose	To verify whether the Charging Station is able to report that its connector is <i>Occupied</i> .	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	1. Execute Reusable State <i>EVConnectedPreSession</i>	
Tool validations	N/a	
	Post scenario validations: N/a	

Table 160. Test Case Id: TC_G_02_CS

Test case name	Connector status Notification - Occupied to Available	
Test case Id	TC_G_02_CS	
Use case Id(s)	G01, N07	
Requirement(s)	G01.FR.01, N07.FR.19	
System under test	Charging Station	
Description	A Charging Station sends a notification to the CSMS to inform the CSMS about a Connector status change. This can be done in two ways. Via a StatusNotificationRequest or a NotifyEventRequest from the device model.	
Purpose	To verify whether the Charging Station is able to report that its connector is Available_.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): State is <i>EVConnectedPreSession</i>	
Main (Test scenario)	Charging Station <u>Manual Action:</u> Disconnect the EV and EVSE. 3. The Charging Station notifies the CSMS about the current state of the connector.	CSMS 4. The OCTT responds accordingly.
Tool validations	* Step 3: Message: StatusNotificationRequest - connectorStatus Available Message: NotifyEventRequest - eventData[0].trigger Delta - eventData[0].actualValue "Available" - eventData[0].component.name "Connector" - eventData[0].variable.name "AvailabilityState" Post scenario validations: N/a	

Table 161. Test Case Id: TC_G_03_CS

Test case name	Change Availability EVSE - Operative to inoperative	
Test case Id	TC_G_03_CS	
Use case Id(s)	G03	
Requirement(s)	G03.FR.01, G03.FR.02, G03.FR.04, G03.FR.06	
System under test	Charging Station	
Description	This test case covers how the CSMS requests the Charging Station to change the availability of one of the EVSEs from Operative to Inoperative. An EVSE is considered Operative in any status other than Faulted and Unavailable.	
Purpose	To verify if the Charging Station is able to perform the change availability mechanism as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	1. Execute Reusable State <i>Unavailable</i> for <Configured evseld>	
Tool validations	N/a	
	Post scenario validations: - A message to report the state of a connector has been received for all connectors belonging to the specified EVSE.	

Table 162. Test Case Id: TC_G_04_CS

Test case name	Change Availability EVSE - Inoperative to operative	
Test case Id	TC_G_04_CS	
Use case Id(s)	G03	
Requirement(s)	G03.FR.01, G03.FR.02, G03.FR.04, G03.FR.07	
System under test	Charging Station	
Description	This test case covers how the CSMS requests the Charging Station to change the availability of one of the EVSEs from Inoperative to Operative. An EVSE is considered Operative in any status other than Faulted and Unavailable.	
Purpose	To verify if the Charging Station is able to perform the change availability mechanism as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	Configuration State: N/a Memory State: <i>Unavailable</i> for <Configured evseld> Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with a ChangeAvailabilityResponse	1. The OCTT sends a ChangeAvailabilityRequest with operationalStatus Operative and evse.id <Configured evseld>
	3. The Charging Station notifies the CSMS about the current state of all connectors belonging to the specified EVSE (and optionally also from the EVSE itself).	4. The OCTT responds accordingly.
Tool validations	<ul style="list-style-type: none"> * Step 2: Message ChangeAvailabilityResponse - status Accepted * Step 3: Message: StatusNotificationRequest - connectorStatus Available - evseld <Configured evseld> Message: NotifyEventRequest - eventData[0].trigger Delta - eventData[0].actualValue "Available" - eventData[0].component.name "EVSE" / Connector - eventData[0].component.evse.id <Configured evseld> - eventData[0].variable.name "AvailabilityState" <p>Post scenario validations: - A message to report the state of a connector has been received for all connectors belonging to the specified EVSE.</p>	

Table 163. Test Case Id: TC_G_05_CS

Test case name	Change Availability Charging Station - Operative to inoperative	
Test case Id	TC_G_05_CS	
Use case Id(s)	G04	
Requirement(s)	G04.FR.01, G04.FR.02, G04.FR.03, G04.FR.05, G04.FR.07	
System under test	Charging Station	
Description	<p>This test case describes how the CSMS requests the Charging Station to change the availability from operative to inoperative.</p> <p>A Charging Station is considered Operative when it is charging or ready for charging.</p> <p>A Charging Station is considered Inoperative when it does not allow any charging.</p>	
Purpose	To verify if the Charging Station is able to perform the change availability mechanism as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	1. Execute Reusable State <i>Unavailable</i>	
Tool validations	N/a Post scenario validations: - A message to report the state of a connector has been received for all connectors.	

Table 164. Test Case Id: TC_G_06_CS

Test case name	Change Availability Charging Station - Inoperative to operative							
Test case Id	TC_G_06_CS							
Use case Id(s)	G04							
Requirement(s)	G04.FR.01, G04.FR.02, G04.FR.03, G04.FR.05, G04.FR.08							
System under test	Charging Station							
Description	<p>This test case describes how the CSMS requests the Charging Station to change the availability from inoperative to operative.</p> <p>A Charging Station is considered Operative when it is charging or ready for charging.</p> <p>A Charging Station is considered Inoperative when it does not allow any charging.</p>							
Purpose	To verify if the Charging Station is able to perform the change availability mechanism as described at the OCPP specification.							
Prerequisite(s)	n/a							
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): State is <i>Unavailable</i>							
Main (Test scenario)	<table border="1"> <tr> <td>Charging Station</td> <td>CSMS</td> </tr> <tr> <td>2. The Charging Station responds with a ChangeAvailabilityResponse</td> <td>1. The OCTT sends a ChangeAvailabilityRequest with operationalStatus Operative</td> </tr> <tr> <td>3. The Charging Station notifies the CSMS about the current state of all its connectors (and optionally also the Charging Station itself and all EVSE).</td> <td>4. The OCTT responds accordingly.</td> </tr> </table>	Charging Station	CSMS	2. The Charging Station responds with a ChangeAvailabilityResponse	1. The OCTT sends a ChangeAvailabilityRequest with operationalStatus Operative	3. The Charging Station notifies the CSMS about the current state of all its connectors (and optionally also the Charging Station itself and all EVSE).	4. The OCTT responds accordingly.	
Charging Station	CSMS							
2. The Charging Station responds with a ChangeAvailabilityResponse	1. The OCTT sends a ChangeAvailabilityRequest with operationalStatus Operative							
3. The Charging Station notifies the CSMS about the current state of all its connectors (and optionally also the Charging Station itself and all EVSE).	4. The OCTT responds accordingly.							
Tool validations	<ul style="list-style-type: none"> * Step 2: Message ChangeAvailabilityResponse - status Accepted * Step 3: Message: StatusNotificationRequest - connectorStatus Available Message: NotifyEventRequest - eventData[0].trigger Delta - eventData[0].actualValue "Available" - eventData[0].component.name "ChargingStation" / EVSE / Connector - eventData[0].variable.name "AvailabilityState" <p>Post scenario validations: - A message to report the state of a connector has been received for all connectors.</p>							

Table 165. Test Case Id: TC_G_07_CS

Test case name	Change Availability Connector - Operative to inoperative	
Test case Id	TC_G_07_CS	
Use case Id(s)	G03	
Requirement(s)	G03.FR.01, G03.FR.02, G03.FR.04, G03.FR.06	
System under test	Charging Station	
Description	This test case covers how the CSMS requests the Charging Station to change the availability of one of the Connectors of one EVSE from Operative to Inoperative. A Connector is considered Operative in any status other than Faulted and Unavailable.	
Purpose	To verify if the Charging Station is able to perform the change availability mechanism as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	1. Execute Reusable State <i>Unavailable</i> for <Configured connectorId>	
Tool validations	N/a	
	Post scenario validations: - A message to report the state of the connector has been received.	

Table 166. Test Case Id: TC_G_08_CS

Test case name	Change Availability Connector - Inoperative to operative	
Test case Id	TC_G_08_CS	
Use case Id(s)	G03	
Requirement(s)	G03.FR.01, G03.FR.02, G03.FR.04, G03.FR.07	
System under test	Charging Station	
Description	This test case covers how the CSMS requests the Charging Station to change the availability of one of the Connectors from one EVSE from Inoperative to Operative. A Connector is considered Operative in any status other than Faulted and Unavailable.	
Purpose	To verify if the Charging Station is able to perform the change availability mechanism as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	Configuration State: N/a Memory State: <i>Unavailable</i> for <Configured connectorId> Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with a ChangeAvailabilityResponse	1. The OCTT sends a ChangeAvailabilityRequest with operationalStatus <i>Operative</i> and evse.id <Configured evseld> and evse.connectorId <Configured connectorId>
	3. The Charging Station notifies the CSMS about the current state of the connectors.	4. The OCTT responds accordingly.
Tool validations	<ul style="list-style-type: none"> * Step 2: Message ChangeAvailabilityResponse <ul style="list-style-type: none"> - status Accepted * Step 3: Message: StatusNotificationRequest <ul style="list-style-type: none"> - connectorStatus Available - evseld <Configured evseld> - connectorId <Configured connectorId> Message: NotifyEventRequest <ul style="list-style-type: none"> - eventData[0].trigger Delta - eventData[0].actualValue "Available" - eventData[0].component.name "Connector" - eventData[0].component.evse.id <Configured evseld> - eventData[0].component.evse.connectorId <Configured connectorId> - eventData[0].variable.name "AvailabilityState" <p>Post scenario validations:</p> <ul style="list-style-type: none"> - A message to report the state of the connector has been received. 	

Table 167. Test Case Id: TC_G_09_CS

Test case name	Change Availability EVSE - Operative to operative	
Test case Id	TC_G_09_CS	
Use case Id(s)	G03	
Requirement(s)	G03.FR.01, G03.FR.02, G03.FR.03, G03.FR.04	
System under test	Charging Station	
Description	This test case covers how the CSMS requests the Charging Station to change the availability of one of the EVSEs from Operative to Operative. An EVSE is considered Operative in any status other than Faulted and Unavailable.	
Purpose	To verify if the Charging Station is able to perform the change availability from Operative to Operative according to the mechanism as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with a ChangeAvailabilityResponse	1. The OCTT sends a ChangeAvailabilityRequest with operationalStatus Operative and evse.id <Configured evselId>
Tool validations	<ul style="list-style-type: none"> * Step 2: Message ChangeAvailabilityResponse - status Accepted 	
	Post scenario validations: N/a	

Table 168. Test Case Id: TC_G_10_CS

Test case name	Change Availability EVSE - Inoperative to inoperative	
Test case Id	TC_G_10_CS	
Use case Id(s)	G03	
Requirement(s)	G03.FR.01, G03.FR.02, G03.FR.03, G03.FR.04	
System under test	Charging Station	
Description	This test case covers how the CSMS requests the Charging Station to change the availability of one of the EVSEs from Inoperative to Inoperative. An EVSE is considered Inoperative in status Faulted and Unavailable.	
Purpose	To verify if the Charging Station is able to perform the change availability from inoperative to inoperative according to the mechanism as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Charging State: State is <i>Unavailable</i> for <Configured evselId>	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with a ChangeAvailabilityResponse	1. The OCTT sends a ChangeAvailabilityRequest with operationalStatus Inoperative and evse.id <Configured evselId>
Tool validations	<ul style="list-style-type: none"> * Step 2: Message ChangeAvailabilityResponse - status Accepted <p>Post scenario validations:</p> <ul style="list-style-type: none"> - A message to report the state of a connector has been received for all connectors. 	

Table 169. Test Case Id: TC_G_11_CS

Test case name	Change Availability EVSE - With ongoing transaction	
Test case Id	TC_G_11_CS	
Use case Id(s)	G03	
Requirement(s)	G03.FR.01, G03.FR.02, G03.FR.04, G03.FR.05	
System under test	Charging Station	
Description	This test case covers how the CSMS requests the Charging Station to change the availability of one of the EVSEs from Operative to Inoperative. An EVSE is considered Operative in any status other than Faulted and Unavailable.	
Purpose	To verify if the Charging Station is able to perform the change availability during a transaction according to the mechanism as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	Configuration State: N/a Memory State: N/a Charging State: State is <i>EnergyTransferStarted</i>	
Main (Test scenario)	Charging Station 2. The Charging Station responds with a ChangeAvailabilityResponse	CSMS 1. The OCTT sends a ChangeAvailabilityRequest with operationalStatus <i>Inoperative</i> and evse.id <Configured evseld>
	Note(s): Wait for <Configured Transaction Duration> 3. Execute Reusable State <i>StopAuthorized</i>	
	4. The Charging Station notifies the CSMS about the current state of the connectors of the configured evse.	5. The OCTT responds accordingly.
	6. Execute Reusable State <i>EVConnectedPostSession</i>	
	7. The Charging Station notifies the CSMS about the current state of the connectors of the configured evse.	8. The OCTT responds accordingly.
	9. Execute Reusable State <i>EVDDisconnected</i>	
	10. The Charging Station notifies the CSMS about the current state of the connectors of the configured evse.	11. The OCTT responds accordingly.
	12. Execute Reusable State <i>ParkingBayUnoccupied</i>	
	13. The Charging Station notifies the CSMS about the current state of the connectors of the configured evse.	14. The OCTT responds accordingly.
	Note(s): Steps 4, 5, 7, 8, 10, 11, 13, and 14 will only be executed if the previous step ended the transaction	
Tool validations	<ul style="list-style-type: none"> * Step 2: Message ChangeAvailabilityResponse - status <i>Scheduled</i> * Step 4, 7, 10, 13: Message: StatusNotificationRequest - connectorStatus <i>Unavailable</i> - evseld <Configured evseld> Message: NotifyEventRequest - eventData[0].trigger <i>Delta</i> - eventData[0].actualValue "Unavailable" - eventData[0].component.name "Connector" - eventData[0].component.evse.id <Configured evseld> - eventData[0].variable.name "AvailabilityState" <p>Post scenario validations: - A message to report the state of a connector has been received for all connectors.</p>	

Table 170. Test Case Id: TC_G_12_CS

Test case name	Change Availability Charging Station - Operative to operative	
Test case Id	TC_G_12_CS	
Use case Id(s)	G04	
Requirement(s)	G04.FR.01, G04.FR.02, G04.FR.03, G04.FR.04, G04.FR.05	
System under test	Charging Station	
Description	<p>This test case describes how the CSMS requests the Charging Station to change the availability from inoperative to operative.</p> <p>A Charging Station is considered Operative when it is charging or ready for charging.</p> <p>A Charging Station is considered Inoperative when it does not allow any charging.</p>	
Purpose	To verify if the Charging Station is able to perform the change availability from operative to operative according to the mechanism as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	Configuration State: N/a Memory State: N/a Charging State: N/a	
Main (Test scenario)	Charging Station 2. The Charging Station responds with a ChangeAvailabilityResponse	CSMS 1. The OCTT sends a ChangeAvailabilityRequest with operationalStatus Operative
Tool validations	<ul style="list-style-type: none"> * Step 2: Message ChangeAvailabilityResponse - status Accepted Post scenario validations: <ul style="list-style-type: none"> - A message to report the state of a connector has been received for all connectors. 	

Table 171. Test Case Id: TC_G_13_CS

Test case name	Change Availability Charging Station - Inoperative to inoperative	
Test case Id	TC_G_13_CS	
Use case Id(s)	G04	
Requirement(s)	G04.FR.01, G04.FR.02, G04.FR.03, G04.FR.04, G04.FR.05	
System under test	Charging Station	
Description	<p>This test case describes how the CSMS requests the Charging Station to change the availability from operative to inoperative.</p> <p>A Charging Station is considered Operative when it is charging or ready for charging.</p> <p>A Charging Station is considered Inoperative when it does not allow any charging.</p>	
Purpose	To verify if the Charging Station is able to perform the change availability from Inoperative to Inoperative according to the mechanism as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Charging State: State is <i>Unavailable</i>	
Main (Test scenario)	Charging Station <ol style="list-style-type: none"> 2. The Charging Station responds with a ChangeAvailabilityResponse 3. The Charging Station notifies the CSMS about the current state of all connectors. 	CSMS <ol style="list-style-type: none"> 1. The OCTT sends a ChangeAvailabilityRequest with operationalStatus Inoperative 4. The OCTT responds accordingly.
Tool validations	<ul style="list-style-type: none"> * Step 2: Message ChangeAvailabilityResponse - status Accepted * Step 3: Message: StatusNotificationRequest - connectorStatus Unavailable <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none"> - eventData[0].trigger Delta - eventData[0].actualValue "Unavailable" - eventData[0].component.name "ChargingStation" - eventData[0].variable.name "AvailabilityState" <p>Post scenario validations:</p> <ul style="list-style-type: none"> - A message to report the state of a connector has been received for all connectors. 	

Table 172. Test Case Id: TC_G_14_CS

Test case name	Change Availability Charging Station - With ongoing transaction	
Test case Id	TC_G_14_CS	
Use case Id(s)	G04	
Requirement(s)	G04.FR.01, G04.FR.02, G04.FR.03, G04.FR.05, G04.FR.06	
System under test	Charging Station	
Description	This test case covers how the CSMS requests the Charging Station to change the availability of one of the EVSEs from Operative to Inoperative. An EVSE is considered Operative in any status other than Faulted and Unavailable.	
Purpose	To verify if the Charging Station is able to perform the change availability during a transaction according to the mechanism as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	Configuration State: N/a Memory State: N/a Charging State: State is <i>EnergyTransferStarted</i>	
Main (Test scenario)	Charging Station 2. The Charging Station responds with a ChangeAvailabilityResponse 3. The Charging Station notifies the CSMS about the current state of the connectors of the EVSEs that do not have an active transaction.	CSMS 1. The OCTT sends a ChangeAvailabilityRequest with operationalStatus <i>Inoperative</i> 4. The OCTT responds accordingly.
Note(s): Wait for <Configured Transaction Duration>		
5. Execute Reusable State <i>StopAuthorized</i>		
6. The Charging Station notifies the CSMS about the current state of the connectors of the configured evse.		
7. The OCTT responds accordingly.		
8. Execute Reusable State <i>EVConnectedPostSession</i>		
9. The Charging Station notifies the CSMS about the current state of the connectors of the configured evse.		
10. The OCTT responds accordingly.		
11. Execute Reusable State <i>EVDisconnected</i>		
12. The Charging Station notifies the CSMS about the current state of the connectors of the configured evse.		
13. The OCTT responds accordingly.		
14. Execute Reusable State <i>ParkingBayUnoccupied</i>		
15. The Charging Station notifies the CSMS about the current state of the connectors of the configured evse.		
16. The OCTT responds accordingly.		
Note(s): Steps 6, 7, 9, 10, 12, 13, 15, and 16 will only be executed if the previous step ended the transaction		

Test case name	Change Availability Charging Station - With ongoing transaction
Tool validations	<ul style="list-style-type: none"> * Step 2: Message ChangeAvailabilityResponse - status <i>Scheduled</i> * Step 7: Message: StatusNotificationRequest - connectorStatus <i>Unavailable</i> - evsId not 0 - connectorId not 0 Message: NotifyEventRequest - eventData[0].trigger <i>Delta</i> - eventData[0].actualValue "Unavailable" - eventData[0].component.name "Connector" - eventData[0].variable.name "AvailabilityState"
	<p>Post scenario validations:</p> <ul style="list-style-type: none"> - A message to report the state of a connector has been received for all connectors.

Table 173. Test Case Id: TC_G_15_CS

Test case name	Change Availability Connector - Operative to operative	
Test case Id	TC_G_15_CS	
Use case Id(s)	G03	
Requirement(s)	G03.FR.01, G03.FR.02, G03.FR.03, G03.FR.04	
System under test	Charging Station	
Description	This test case covers how the CSMS requests the Charging Station to change the availability of one of the EVSEs from Operative to Inoperative. An EVSE is considered Operative in any status other than Faulted and Unavailable.	
Purpose	To verify if the Charging Station is able to perform the change availability from Operative to Operative of one connector according to the mechanism as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with a ChangeAvailabilityResponse	1. The OCTT sends a ChangeAvailabilityRequest with operationalStatus Operative and evse.id <Configured evselId> and evse.connectorId <Configured connectorId>
Tool validations	<ul style="list-style-type: none"> * Step 2: Message ChangeAvailabilityResponse - status Accepted <p>Post scenario validations:</p> <ul style="list-style-type: none"> - A message to report the state of a connector has been received for all connectors. 	

Table 174. Test Case Id: TC_G_16_CS

Test case name	Change Availability Connector - Inoperative to inoperative	
Test case Id	TC_G_16_CS	
Use case Id(s)	G03	
Requirement(s)	G03.FR.01, G03.FR.02, G03.FR.03, G03.FR.04	
System under test	Charging Station	
Description	This test case covers how the CSMS requests the Charging Station to change the availability of one of the EVSEs from Operative to Inoperative. An EVSE is considered Operative in any status other than Faulted and Unavailable.	
Purpose	To verify if the Charging Station is able to perform the change availability from inoperative to inoperative on one connector according to the mechanism as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Charging State: State is <i>Unavailable</i>	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with a ChangeAvailabilityResponse	1. The OCTT sends a ChangeAvailabilityRequest with operationalStatus Inoperative and evse.id <Configured evseld> and evse.connectorId <Configured connectorId>
Tool validations	<ul style="list-style-type: none"> * Step 2: Message ChangeAvailabilityResponse - status Accepted <p>Post scenario validations:</p> <ul style="list-style-type: none"> - A message to report the state of a connector has been received for all connectors. 	

Table 175. Test Case Id: TC_G_17_CS

Test case name	Change Availability Connector - With ongoing transaction	
Test case Id	TC_G_17_CS	
Use case Id(s)	G03	
Requirement(s)	G03.FR.01, G03.FR.02, G03.FR.04, G03.FR.05	
System under test	Charging Station	
Description	This test case covers how the CSMS requests the Charging Station to change the availability of one of the EVSEs from Operative to Inoperative. An EVSE is considered Operative in any status other than Faulted and Unavailable.	
Purpose	To verify if the Charging Station is able to perform the change availability during a transaction according to the mechanism as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	<p>Configuration State: N/a</p> <p>Memory State: N/a</p> <p>Charging State: State is <i>EnergyTransferStarted</i></p>	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with a ChangeAvailabilityResponse	1. The OCTT sends a ChangeAvailabilityRequest with operationalStatus <i>Inoperative</i> and evse.id <Configured evselId> and evse.connectorId <Configured connectorId>
	<u>Note(s):</u> Wait for <Configured Transaction Duration>	
	3. Execute Reusable State <i>StopAuthorized</i>	
	4. The Charging Station notifies the CSMS about the current state of the connectors of the configured evse.	5. The OCTT responds accordingly.
	6. Execute Reusable State <i>EVConnectedPostSession</i>	
	7. The Charging Station notifies the CSMS about the current state of the connectors of the configured evse.	8. The OCTT responds accordingly.
	9. Execute Reusable State <i>EVDisconnected</i>	
	10. The Charging Station notifies the CSMS about the current state of the connectors of the configured evse.	11. The OCTT responds accordingly.
	12. Execute Reusable State <i>ParkingBayUnoccupied</i>	
	13. The Charging Station notifies the CSMS about the current state of the connectors of the configured evse.	14. The OCTT responds accordingly.
	<u>Note(s):</u> Steps 4, 5, 7, 8, 10, 11, 13, and 14 will only be executed if the previous step ended the transaction	

Test case name	Change Availability Connector - With ongoing transaction
Tool validations	<ul style="list-style-type: none"> * Step 2: Message ChangeAvailabilityResponse - status <i>Scheduled</i> * Step 7: Message: StatusNotificationRequest - connectorStatus <i>Unavailable</i> - evselId <Configured evselId> - connectorId <Configured connectorId> Message: NotifyEventRequest - eventData[0].trigger <i>Delta</i> - eventData[0].actualValue "Unavailable" - eventData[0].component.name "Connector" - eventData[0].component.evse not <i>omit</i> - eventData[0].component.evse.id <Configured evselId> - eventData[0].component.evse.connectorId <Configured connectorId> - eventData[0].variable.name "AvailabilityState"
	<p>Post scenario validations:</p> <ul style="list-style-type: none"> - A message to report the state of a connector has been received for all connectors.

Table 176. Test Case Id: TC_G_18_CS

Test case name	Change Availability EVSE - state persists across reboot	
Test case Id	TC_G_18_CS	
Use case Id(s)	G03	
Requirement(s)	G03.FR.08. G01.FR.01	
System under test	Charging Station	
Description	This test case covers how the CSMS requests the Charging Station to change the availability of one of the EVSEs from Operative to Inoperative. An EVSE is considered Operative in any status other than Faulted and Unavailable.	
Purpose	To verify if the Charging Station sets the availability persistent across reboot/power loss as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	<p>Configuration State: N/a</p> <p>Memory State: N/a</p> <p>Charging State: state is <i>Unavailable</i></p>	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with a ChangeAvailabilityResponse	1. The OCTT sends a ChangeAvailabilityRequest with operationalStatus <i>Inoperative</i> AND evse.id <Configured evsId>
	3. The Charging Station notifies the CSMS about the current state of all connectors.	4. The OCTT responds accordingly.
	5. Execute Reusable State Booted	
	<p>Note(s):</p> <ul style="list-style-type: none"> - After booting the charging station should send the following status: <p>Message: StatusNotificationRequest</p> <ul style="list-style-type: none"> - connectorStatus <i>Unavailable</i> - evsId <Configured evsId> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none"> - eventData[0].trigger <i>Delta</i> - eventData[0].actualValue "<i>Unavailable</i>" - eventData[0].component.name "<i>Connector</i>" - eventData[0].component.evse.id <Configured evsId> - eventData[0].variable.name "<i>AvailabilityState</i>" 	
Tool validations	<ul style="list-style-type: none"> * Step 2: Message ChangeAvailabilityResponse - status <i>Accepted</i> * Step 3: Message: StatusNotificationRequest - evsId not 0 - connectorId not 0 - connectorStatus <i>Unavailable</i> for evsId <Configured evsId> - connectorStatus <i>Available</i> for evsId not <Configured evsId> Message: NotifyEventRequest - eventData[0].actualValue <i>Unavailable</i> for evsId <Configured evsId> - eventData[0].actualValue <i>Available</i> for evsId not <Configured evsId> <p>Post scenario validations:</p> <ul style="list-style-type: none"> - A message to report the state of a connector has been received for all connectors. 	

Table 177. Test Case Id: TC_G_19_CS

Test case name	Change Availability Connector - state persists across reboot	
Test case Id	TC_G_19_CS	
Use case Id(s)	G03	
Requirement(s)	G03.FR.08	
System under test	Charging Station	
Description	This test case covers how the CSMS requests the Charging Station to change the availability of one of the Connectors of one EVSE from Operative to Inoperative. A Connector is considered Operative in any status other than Faulted and Unavailable.	
Purpose	To verify if the Charging Station sets the availability persistent across reboot/power loss as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	Configuration State: N/a Memory State: N/a Charging State: state is <i>Unavailable</i>	
Main (Test scenario)	Charging Station	CSMS
	1. Execute Reusable State Booting 2. The Charging Station sends a BootNotificationRequest 3. The OCTT responds with a BootNotificationResponse . 4. The Charging Station reports the status of all its connectors. 5. The OCTT responds accordingly. 6. The Charging Station sends a SecurityEventNotificationRequest 7. The OCTT responds with a SecurityEventNotificationResponse	
Tool validations	<ul style="list-style-type: none"> * Step 4: Message: StatusNotificationRequest <ul style="list-style-type: none"> - evselId not 0 - connectorId not 0 - connectorStatus <i>Unavailable</i> for evselId <Configured evselId> and for connectorId <Configured ConnectorId> - connectorStatus <i>Available</i> for evselId not <Configured evselId> and for connectorId <Configured ConnectorId> Message: NotifyEventRequest <ul style="list-style-type: none"> - eventData[0].actualValue <i>Unavailable</i> for evselId <Configured evselId> and for connectorId <Configured ConnectorId> - eventData[0].actualValue <i>Available</i> for evselId not <Configured evselId> and for connectorId <Configured ConnectorId> * Step 6: Message: SecurityEventNotificationRequest <ul style="list-style-type: none"> - type "StartupOfTheDevice" or type "ResetOrReboot" Post scenario validations: <ul style="list-style-type: none"> - A message to report the state of a connector has been received for all connectors. 	

Table 178. Test Case Id: TC_G_21_CS

Test case name	Change Availability Charging Station - state persists across reboot	
Test case Id	TC_G_21_CS	
Use case Id(s)	G04	
Requirement(s)	G04.FR.09	
System under test	Charging Station	
Description	This test case covers how the CSMS requests the Charging Station to change the availability from Operative to Inoperative. An EVSE is considered Operative in any status other than Faulted and Unavailable.	
Purpose	To verify if the Charging Station sets the availability persistent across reboot/power loss as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	Configuration State: N/a Memory State: N/a Charging State: State is <i>Unavailable</i>	
Main (Test scenario)	Charging Station	CSMS
	1. Execute Reusable State Booting	
	2. The Charging Station sends a BootNotificationRequest.	3. The OCTT responds with a BootNotificationResponse.
	4. The Charging Station reports the status of all its connectors.	5. The OCTT responds accordingly.
	6. The Charging Station sends a SecurityEventNotificationRequest	7. The OCTT responds with a SecurityEventNotificationResponse
Tool validations	* Step 4: Message: StatusNotificationRequest - evsId not 0 - connectorId not 0 - connectorStatus <i>Unavailable</i> Message: NotifyEventRequest - eventData[0].trigger <i>Delta</i> - eventData[0].actualValue " <i>Unavailable</i> " - eventData[0].variable.name " <i>AvailabilityState</i> " * Step 6: Message: SecurityEventNotificationRequest - type " <i>StartupOfTheDevice</i> " or type " <i>ResetOrReboot</i> " Post scenario validations: - A message to report the state of a connector has been received for all connectors.	

2.9. H Reservation

This section is intentionally blank, this will be added in a later version.

2.10. I Tariff and Cost

This section is intentionally blank, this will be added in a later version.

2.11. J MeterValues

Table 179. Test Case Id: TC_J_01_CS

Test case name	Clock-aligned Meter Values - No transaction ongoing	
Test case Id	TC_J_01_CS	
Use case Id(s)	J01	
Requirement(s)	J01.FR.01,J01.FR.02,J01.FR.03,J01.FR.06,J01.FR.07,J01.FR.08,J01.FR.14,J01.FR.15	
System under test	Charging Station	
Description	The Charging Station samples the electrical meter or other sensor/transducer hardware to provide information about its Meter Values. Depending on configuration settings, the Charging Station will send Meter Values.	
Purpose	To verify if the Charging Station is able to send clock-aligned Meter Values, when it is configured to do so.	
Prerequisite(s)	The Charging Station has an energy meter.	
Before (Preparations)	Configuration State: AlignedDataInterval is <Configured clock-aligned Meter Values interval> Memory State: N/a Reusable State(s): N/a	
Main (Test scenario)	Charging Station 1. The Charging Station notifies the CSMS about its measured Meter Values. <u>Note(s):</u> - The Meter Value messages do NOT need to be send/received at the exact specified interval. The configured measurands must be measured at the configured interval. - Multiple Meter Value messages may be sent per configured interval. One (or more in case the amount of measured data is too much for one message) for each EVSE and one (or more) for the main power meter (connectorId=0) - The OCTT will end the testcase after it has received three Meter Value messages.	CSMS 2. The OCTT responds accordingly.

Test case name	Clock-aligned Meter Values - No transaction ongoing
Tool validations	<ul style="list-style-type: none"> * Step 1: <p>Message: MeterValuesRequest</p> <ul style="list-style-type: none"> - sampledValue[0].context must be <i>Sample.Clock</i> - sampledValue must contain <i><An element per configured measurand at the AlignedDataMeasurands. The measurand field may be omitted when the measurand is "Energy.Active.Import.Register"></i> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none"> - eventData must contain <i><An element per configured measurand at the AlignedDataMeasurands.></i> - trigger must be <i>Periodic</i> - component.name must be <i>"FiscalMetering"</i> <p><i>Note: The following tool validation will NOT be validated by the OCTT:</i></p> <ul style="list-style-type: none"> - variable.name must <i><Refer to the configured measurand in PascalCase without a "." in between. For example; "EnergyActiveImportRegister"></i> <p>Post scenario validations:</p> <p>Message: MeterValuesRequest</p> <ul style="list-style-type: none"> - timestamp <i><The intervals between the timestamps of the received Meter Value messages must equal the configured value at AlignedDataInterval. However it is allowed to send multiple Meter Value messages per configured interval. One (or more in case the amount of measured data is too much for one message) for each EVSE and one (or more) for the main power meter (evseld=0). But the timestamp of these messages must all be the same.></i> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none"> - timestamp <i><The intervals between the timestamps of the received Meter Value messages must equal the configured value at AlignedDataInterval. However it is allowed to send multiple Meter Value messages per configured interval. One (or more in case the amount of measured data is too much for one message) for each EVSE and one (or more) for the main power meter (evseld=0). But the timestamp of these messages must all be the same.></i>

Table 180. Test Case Id: TC_J_02_CS

Test case name	Clock-aligned Meter Values - Transaction ongoing	
Test case Id	TC_J_02_CS	
Use case Id(s)	J01	
Requirement(s)	J01.FR.01,J01.FR.02,J01.FR.03,J01.FR.06,J01.FR.07,J01.FR.08,J01.FR.14,J01.FR.15	
System under test	Charging Station	
Description	The Charging Station samples the electrical meter or other sensor/transducer hardware to provide information about its Meter Values. Depending on configuration settings, the Charging Station will send Meter Values.	
Purpose	To verify if the Charging Station is able to send clock-aligned Meter Values, while a transaction is ongoing, when it is configured to do so.	
Prerequisite(s)	The Charging Station has an energy meter.	
Before (Preparations)	<p>Configuration State: AlignedDataInterval is <Configured clock-aligned Meter Values interval> AlignedDataSendDuringIdle is <i>false</i> (If implemented)</p> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>EnergyTransferStarted</i></p>	
Main (Test scenario)	<p>Charging Station</p> <p><u>Note(s):</u></p> <ul style="list-style-type: none"> - The Charging Station can follow Steps 1 and 2 or Steps 3 and 4 <p>1. The Charging Station notifies the CSMS about its measured Meter Values.</p> <p><u>Note(s):</u></p> <ul style="list-style-type: none"> - During a transaction the MeterValueRequest or NotifyEventRequest can still be used to report meter values for the main power meter (evsId=0) and idle EVSEs - The Meter Value messages do NOT need to be send/received at the exact specified interval. The configured measurands must be measured at the configured interval. - Multiple Meter Value messages may be sent per configured interval, in case the amount of measured data is too much for one message. <p>3. The Charging Station sends a TransactionEventRequest</p> <p><u>Note(s):</u></p> <ul style="list-style-type: none"> - During a transaction the meter values for the configured EVSE with the ongoing transaction should be transmitted using the TransactionEventRequest. - The TransactionEventRequest messages do NOT need to be send/received at the exact specified interval. The configured measurands must be measured at the configured interval. - Multiple TransactionEventRequest messages may be sent per configured interval, in case the amount of measured data is too much for one message. - The OCTT will end the testcase after it has the <Configured transaction duration> is reached.. 	<p>CSMS</p> <p>2. The OCTT responds accordingly.</p> <p>4. The OCTT responds with a TransactionEventResponse</p>

Test case name	Clock-aligned Meter Values - Transaction ongoing
Tool validations	<p><i>Note: The following steps do not need to be sent in a specific order.</i></p> <p>* Step 1:</p> <p>Message: MeterValuesRequest</p> <ul style="list-style-type: none"> - meterValue[0].sampledValue[0].context must be <code>Sample.Clock</code> - meterValue[0].sampledValue must contain <code><An element per configured measurand at the AlignedDataMeasurands. The measurand field may be omitted when the measurand is "Energy.Active.Import.Register"></code> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none"> - eventData must contain <code><An element per configured measurand at the AlignedDataMeasurands.></code> - trigger must be <code>Periodic</code> - component.name must be <code>"FiscalMetering"</code> <p><i>Note: The following tool validation will NOT be validated by the OCTT:</i></p> <ul style="list-style-type: none"> - variable.name must <code><Refer to the configured measurand in PascalCase without a "." in between. For example; "EnergyActiveImportRegister"></code> <p>* Step 3:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - triggerReason must be <code>MeterValueClock</code> - metervalue[0].sampledValue[0].context must be <code>Sample.Clock</code> - metervalue[0].sampledValue must contain <code><An element per configured measurand at the AlignedDataMeasurands. The measurand field may be omitted when the measurand is "Energy.Active.Import.Register"></code>
Post scenario validations:	<p>Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - timestamp <i><The intervals between the timestamps of the received TransactionEventRequest messages must equal the configured value at AlignedDataInterval. However it is allowed to send multiple Meter Value messages per configured interval, in case the amount of measured data is too much for one message. But the timestamp of these messages must all be the same.></i> <p>Message: MeterValuesRequest</p> <ul style="list-style-type: none"> - timestamp <i><The intervals between the timestamps of the received Meter Value messages must equal the configured value at AlignedDataInterval. However it is allowed to send multiple Meter Value messages per configured interval, in case the amount of measured data is too much for one message. But the timestamp of these messages must all be the same.></i> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none"> - timestamp <i><The intervals between the timestamps of the received Meter Value messages must equal the configured value at AlignedDataInterval. However it is allowed to send multiple Meter Value messages per configured interval, in case the amount of measured data is too much for one message. But the timestamp of these messages must all be the same.></i>

Table 181. Test Case Id: TC_J_03_CS

Test case name	Clock-aligned Meter Values - EventType Ended	
Test case Id	TC_J_03_CS	
Use case Id(s)	J01 & (E06,E07,E08,E09,E10,E12)	
Requirement(s)	J01.FR.01,J01.FR.02,J01.FR.03,J01.FR.06,J01.FR.07,J01.FR.08,J01.FR.14,J01.FR.15 & E06.FR.11,E06.FR.17,E07.FR.08,E07.FR.13,E08.FR.09,E09.FR.05,E10.FR.04,E12.FR.07	
System under test	Charging Station	
Description	The Charging Station samples the electrical meter or other sensor/transducer hardware to provide information about its Meter Values. Depending on configuration settings, the Charging Station will send Meter Values.	
Purpose	To verify if the Charging Station is able to send clock-aligned Meter Values when a transaction ends at the TransactionEventRequest with eventType is <i>Ended</i> , when it is configured to do so.	
Prerequisite(s)	The Charging Station has an energy meter.	
Before (Preparations)	<p>Configuration State: AlignedDataTxEndedInterval is <Configured clock_aligned_tx-ended_meter_values_interval> AlignedDataSendDuringIdle is false (If implemented)</p> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>EnergyTransferStarted</i></p>	
Main (Test scenario)	<p>Charging Station</p> <p>1. Execute Reusable State <i>ParkingBayUnoccupied</i></p> <p><u>Note(s):</u></p> <ul style="list-style-type: none"> - This step will be executed after the <Configured transaction duration> is reached. - This causes the transaction to stop. 	CSMS
Tool validations	<p>N/a</p> <p>Post scenario validations:</p> <ul style="list-style-type: none"> - The TransactionEventRequest containing eventType <i>Ended</i> contains the MeterValue field. - The MeterValue must contain <An element per data collection moment indicated by AlignedDataTxEndedInterval. The OCTT will not validate this.> - timestamp <The intervals between the timestamps of the received Meter Value messages must equal the configured value at AlignedDataTxEndedInterval.> - sampledValue[0].context must be <i>Sample.Clock</i> - sampledValue must contain <An element per configured measurand at the AlignedDataTxEndedMeasurands. The measurand field may be omitted when the measurand is "Energy.Active.Import.Register"> 	

Table 182. Test Case Id: TC_J_04_CS

Test case name	Clock-aligned Meter Values - Signed	
Test case Id	TC_J_04_CS	
Use case Id(s)	J01	
Requirement(s)	J01.FR.21	
System under test	Charging Station	
Description	The Charging Station samples the electrical meter or other sensor/transducer hardware to provide information about its Meter Values. Depending on configuration settings, the Charging Station will send Meter Values.	
Purpose	To verify if the Charging Station is able to send signed clock-aligned Meter Values when a transaction ends at the TransactionEventRequest with eventType is <i>Ended</i> , when it is configured to do so.	
Prerequisite(s)	The Charging Station has an energy meter.	
Before (Preparations)	<p>Configuration State: AlignedDataTxEndedInterval is <Configured clock_aligned_tx_ended_meter_values_interval> AlignedDataSendDuringIdle is <i>false</i> (If implemented) AlignedDataSignReadings is <i>true</i></p> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>EnergyTransferStarted</i></p>	
Main (Test scenario)	<p>Charging Station</p> <p>1. Execute Reusable State <i>ParkingBayUnoccupied</i></p> <p>Note(s):</p> <ul style="list-style-type: none"> - This step will be executed after the <Configured transaction duration> is reached. - This causes the transaction to stop. 	CSMS
Tool validations	<p>N/a</p> <p>Post scenario validations:</p> <ul style="list-style-type: none"> - The TransactionEventRequest containing eventType <i>Ended</i> contains the MeterValue field. - The MeterValue should contain <An element per data collection moment indicated by AlignedDataTxEndedInterval. The OCTT will not validate this.> - timestamp <The intervals between the timestamps of the received Meter Value messages should equal the configured value at AlignedDataTxEndedInterval.> - sampledValue[0].context should be <i>Sample.Clock</i> - sampledValue should contain <An element per configured measurand at the AlignedDataTxEndedMeasurands. The measurand field may be omitted when the measurand is "Energy.Active.Import.Register"> - sampledValue.signedMeterValue should not be omitted - sampledValue.signedMeterValue.publicKey should exist and depending on the value of OCPPCommCtrlr.PublicKeyWithSignedMeterValue, should be either "", or a valid public key 	

Table 183. Test Case Id: TC_J_06_CS

Test case name	Clock-aligned Meter Values - No Meter Values during transaction	
Test case Id	TC_J_06_CS	
Use case Id(s)	J01	
Requirement(s)	N/a	
System under test	Charging Station	
Description	The Charging Station samples the electrical meter or other sensor/transducer hardware to provide information about its Meter Values. Depending on configuration settings, the Charging Station will send Meter Values.	
Purpose	To verify if the Charging Station is able to only send clock-aligned Meter Values when there is no ongoing transaction, when it is configured to do so.	
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station has an energy meter. - The configuration variable <code>AlignedDataSendDuringIdle</code> is implemented. 	
Before (Preparations)	<p>Configuration State: AlignedDataInterval is set to <Configured clock-aligned Meter Values interval> AlignedDataSendDuringIdle is set to <i>true</i></p> <p>Memory State: N/a</p> <p>Reusable State(s): N/a</p>	
Main (Test scenario)	<p>Charging Station</p> <p>1. The Charging Station notifies the CSMS about its measured Meter Values.</p> <p>Note(s):</p> <ul style="list-style-type: none"> - The Meter Value messages do NOT need to be send/received at the exact specified interval. The configured measurands must be measured at the configured interval. - Multiple Meter Value messages may be sent per configured interval. One (or more in case the amount of measured data is too much for one message) for each EVSE and one (or more) for the main power meter (<code>evsId=0</code>) <p>3. Execute Reusable State <i>EnergyTransferStarted</i></p> <p>4. The Charging Station notifies the CSMS about its measured Meter Values.</p> <p>Note(s):</p> <ul style="list-style-type: none"> - The Meter Value messages should not be send/received at the exact specified interval. <p>6. Execute Reusable State <i>ParkingBayUnoccupied</i></p> <p>Note(s):</p> <ul style="list-style-type: none"> - This step will be executed after the <Configured clock-aligned Meter Values interval + 5 seconds> is reached. 	<p>CSMS</p> <p>2. The OCTT responds accordingly.</p> <p>5. The OCTT responds accordingly.</p>

Test case name	Clock-aligned Meter Values - No Meter Values during transaction
	<p>7. The Charging Station notifies the CSMS about its measured Meter Values.</p> <p>Note(s):</p> <ul style="list-style-type: none"> - The Meter Value messages do NOT need to be send/received at the exact specified interval. The configured measurands must be measured at the configured interval. - Multiple Meter Value messages may be sent per configured interval. One (or more in case the amount of measured data is too much for one message) for each EVSE and one (or more) for the main power meter (evsId=0) <p>8. The OCTT responds accordingly.</p>
Tool validations	<p>* Step 1 & 7:</p> <p>Message: MeterValuesRequest</p> <ul style="list-style-type: none"> - samplingValue[0].context must be <i>Sample.Clock</i> - samplingValue must contain <An element per configured measurand at the AlignedDataMeasurands. The measurand field may be omitted when the measurand is "Energy.Active.Import.Register"> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none"> - eventData must contain <An element per configured measurand at the AlignedDataMeasurands.> - trigger must be <i>Periodic</i> - component.name must be "FiscalMetering" <p><i>Note: The following tool validation will NOT be validated by the OCTT:</i></p> <ul style="list-style-type: none"> - variable.name must <Refer to the configured measurand in PascalCase without a "." in between. For example; "EnergyActiveImportRegister"> <p>Post scenario validations:</p> <p>Message: MeterValuesRequest</p> <ul style="list-style-type: none"> - timestamp <The intervals between the timestamps of the received Meter Value messages must equal the configured value at AlignedDataInterval. However it is allowed to send multiple Meter Value messages per configured interval. One (or more in case the amount of measured data is too much for one message) for each EVSE and one (or more) for the main power meter (evsId=0). But the timestamp of these messages must all be the same.> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none"> - timestamp <The intervals between the timestamps of the received Meter Value messages must equal the configured value at AlignedDataInterval. However it is allowed to send multiple Meter Value messages per configured interval. One (or more in case the amount of measured data is too much for one message) for each EVSE and one (or more) for the main power meter (evsId=0). But the timestamp of these messages must all be the same.> <p>- The Charging Station did not send any message to report Meter Values to the OCTT, during the time the transaction was active at step 3 and 4. This means none of the following; MeterValuesRequest, NotifyEventRequest for component FiscalMetering / variable (one of the measurand values) OR TransactionEventRequest containing the MeterValue field.</p>

Table 184. Test Case Id: TC_J_07_CS

Test case name	Sampled Meter Values - EventType Started - EVSE known	
Test case Id	TC_J_07_CS	
Use case Id(s)	J02 & (E01,E02,E03,E09,E04,E05)	
Requirement(s)	J02.FR.01,J02.FR.02,J02.FR.03,J02.FR.04,J02.FR.10,E01.FR.09,E02.FR.09,E03.FR.07,E04.FR.05,E05.FR.05	
System under test	Charging Station	
Description	The Charging Station samples the electrical meter or other sensor/transducer hardware to provide information about its Meter Values. Depending on configuration settings, the Charging Station will send Meter Values.	
Purpose	To verify if the Charging Station is able to send sampled Meter Values when a transaction starts and the EVSE is known, at the TransactionEventRequest with eventType is <i>Started</i> , when it is configured to do so.	
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station has an energy meter. - The Charging Station does NOT have the following configuration; TxStartPoint contains <i>ParkingBayOccupancy</i> 	
Before (Preparations)	<p>Configuration State: TxStartPoint contains <i>EVConnected</i> Note: <i>TxStartPoint</i> contains <i>EVConnected</i>, <i>Authorized</i>, <i>PowerPathClosed</i>, <i>EnergyTransfer</i> AND/OR <i>DataSigned</i> (At least one of these values must be set).</p> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>ParkingBayOccupied</i></p>	
Main (Test scenario)	Charging Station	CSMS
	1. Execute Reusable State <i>EVConnectedPreSession</i>	
	2. Execute Reusable State <i>EnergyTransferStarted</i>	
Tool validations	N/a	
	<p>Post scenario validations:</p> <ul style="list-style-type: none"> - The TransactionEventRequest containing <i>eventType Started</i> contains the <i>MeterValue</i> field. - sampledValue[0].context must be <i>Transaction.Begin</i> - sampledValue must contain <An element per configured measurand at the <i>SampledDataTxStartedMeasurands</i>. The <i>measurand</i> field may be omitted when the <i>measurand</i> is "Energy.Active.Import.Register"> 	

Table 185. Test Case Id: TC_J_08_CS

Test case name	Sampled Meter Values - Context Transaction.Begin - EVSE not known	
Test case Id	TC_J_08_CS	
Use case Id(s)	J02 & (E01,E02,E03,E09,E04,E05)	
Requirement(s)	J02.FR.01, J02.FR.02, J02.FR.03, J02.FR.04, J02.FR.10, E01.FR.17, E03.FR.11, E04.FR.11, E05.FR.08	
System under test	Charging Station	
Description	The Charging Station samples the electrical meter or other sensor/transducer hardware to provide information about its Meter Values. Depending on configuration settings, the Charging Station will send Meter Values.	
Purpose	To verify if the Charging Station is able to send sampled Meter Values when a transaction starts and the EVSE is NOT known, NOT at the TransactionEventRequest with eventType is <i>Started</i> , but with eventType <i>Updated</i> , after the EVSE is known and it is configured to do so.	
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station has an energy meter. - The Charging Station does NOT have the following configuration; TxStartPoint does NOT contain <i>ParkingBayOccupancy</i> OR <i>Authorized</i>. - Test case is only applicable when the Charging Station has more than 1 EVSE. 	
Before (Preparations)	<p>Configuration State: TxStartPoint contains <i>Authorized</i> Note: TxStartPoint contains <i>Authorized</i> AND/OR <i>ParkingBayOccupancy</i> (At least one of these values must be set).</p> <p>Memory State: N/a</p> <p>Reusable State(s): N/a</p>	
Main (Test scenario)	Charging Station	CSMS
	1. Execute Reusable State <i>EnergyTransferStarted</i>	
Tool validations	<p>N/a</p> <p>Post scenario validations:</p> <ul style="list-style-type: none"> - The TransactionEventRequest containing eventType <i>Updated</i>, sent during the execution of reusable state <i>EVConnectedPreSession</i> contains the <i>MeterValue</i> field. - sampledValue[0].context must be <i>Transaction.Begin</i> - sampledValue must contain <An element per configured measurand at the <i>SampledDataTxStartedMeasurands</i>. The <i>measurand</i> field may be omitted when the <i>measurand</i> is "Energy.Active.Import.Register"> 	

Table 186. Test Case Id: TC_J_09_CS

Test case name	Sampled Meter Values - EventType Updated	
Test case Id	TC_J_09_CS	
Use case Id(s)	J02 & (E01,E02,E03,E09,E04,E05)	
Requirement(s)	J02.FR.01, J02.FR.02, J02.FR.03, J02.FR.04, J02.FR.10, J02.FR.11, J02.FR.14, E02.FR.10, E02.FR.11, E03.FR.08, E03.FR.09, E04.FR.06, E04.FR.09, E11.FR.03, E11.FR.06, E12.FR.03, E12.FR.06	
System under test	Charging Station	
Description	The Charging Station samples the electrical meter or other sensor/transducer hardware to provide information about its Meter Values. Depending on configuration settings, the Charging Station will send Meter Values.	
Purpose	To verify if the Charging Station is able to send sampled Meter Values during the transaction, at the TransactionEventRequest with eventType is <i>Updated</i> , when it is configured to do so.	
Prerequisite(s)	The Charging Station has an energy meter.	
Before (Preparations)	<p>Configuration State: SampledDataTxUpdatedInterval is <Configured sampled Meter Values Updated interval></p> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>EnergyTransferStarted</i></p>	
Main (Test scenario)	<p>Charging Station</p> <p>1. The Charging Station sends a TransactionEventRequest</p> <p>Note(s):</p> <ul style="list-style-type: none"> - The <i>TransactionEventRequest</i> messages do NOT need to be send/received at the exact specified interval. The configured measurands must be measured at the configured interval. - Multiple <i>TransactionEventRequest</i> messages may be sent per configured interval, in case the amount of measured data is too much for one message. - The OCTT will end the testcase after it has the <Configured transaction duration> is reached.. 	<p>CSMS</p> <p>2. The OCTT responds with a TransactionEventResponse</p>
Tool validations	<p>* Step 1:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - triggerReason must be <i>MeterValuePeriodic</i> - sampledValue[0].context must be <i>Sample.Periodic</i> - sampledValue must contain <An element per configured measurand at the SampledDataTxUpdatedMeasurands. The measurand field may be omitted when the measurand is "Energy.Active.Import.Register"> <p>Post scenario validations:</p> <ul style="list-style-type: none"> - timestamp <The intervals between the timestamps of the received <i>TransactionEventRequest</i> messages must equal the configured value at <i>SampledDataTxUpdatedInterval</i>. However it is allowed to send multiple Meter Value messages per configured interval, in case the amount of measured data is too much for one message. But the timestamp of these messages must all be the same.> 	

Table 187. Test Case Id: TC_J_10_CS

Test case name	Sampled Meter Values - EventType Ended	
Test case Id	TC_J_10_CS	
Use case Id(s)	J02 & (E06,E07,E08,E09,E10,E12)	
Requirement(s)	J02.FR.01,J02.FR.02,J02.FR.03,J02.FR.04,J02.FR.10,E06.FR.11,E06.FR.17, E07.FR.08,E07.FR.13,E08.FR.09,E09.FR.05,E10.FR.04,E12.FR.07	
System under test	Charging Station	
Description	The Charging Station samples the electrical meter or other sensor/transducer hardware to provide information about its Meter Values. Depending on configuration settings, the Charging Station will send Meter Values.	
Purpose	To verify if the Charging Station is able to send sampled Meter Values when a transaction ends at the TransactionEventRequest with eventType is <i>Ended</i> , when it is configured to do so.	
Prerequisite(s)	The Charging Station has an energy meter.	
Before (Preparations)	<p>Configuration State: SampledDataTxEndedInterval is <Configured sampled_tx_ended_meter_values_interval></p> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>EnergyTransferStarted</i></p>	
Main (Test scenario)	<p>Charging Station</p> <p>1. Execute Reusable State <i>ParkingBayUnoccupied</i></p> <p><u>Note(s):</u></p> <ul style="list-style-type: none"> - This step will be executed after the <Configured transaction duration> is reached. - This causes the transaction to stop. 	CSMS
Tool validations	<p>N/a</p> <p>Post scenario validations:</p> <ul style="list-style-type: none"> - The TransactionEventRequest containing eventType <i>Ended</i> contains the MeterValue field. - The MeterValue must contain <An element per data collection moment indicated by SampledDataTxEndedInterval. The OCTT will not validate this.> - timestamp <The intervals between the timestamps of the received Meter Value messages must equal the configured value at SampledDataTxEndedInterval.> - sampledValue[0].context must be <i>Sample.Periodic</i> AND one must have <i>Transaction.End</i> - sampledValue must contain <An element per configured measurand at the SampledDataTxEndedMeasurands. The measurand field may be omitted when the measurand is "Energy.Active.Import.Register"> 	

Table 188. Test Case Id: TC_J_11_CS

Test case name	Sampled Meter Values - Signed	
Test case Id	TC_J_11_CS	
Use case Id(s)	J02	
Requirement(s)	J02.FR.21	
System under test	Charging Station	
Description	The Charging Station samples the electrical meter or other sensor/transducer hardware to provide information about its Meter Values. Depending on configuration settings, the Charging Station will send Meter Values.	
Purpose	To verify if the Charging Station is able to send sampled Meter Values when a transaction ends at the TransactionEventRequest with eventType is <i>Ended</i> , when it is configured to do so.	
Prerequisite(s)	The Charging Station has an energy meter.	
Before (Preparations)	<p>Configuration State: SampledDataTxEndedInterval is <Configured sampled_tx_ended_meter_values_interval> SampledDataSignReadings is true</p> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>EnergyTransferStarted</i></p>	
Main (Test scenario)	Charging Station	CSMS
	<p>1. Execute Reusable State <i>ParkingBayUnoccupied</i></p> <p><u>Note(s):</u></p> <ul style="list-style-type: none"> - This step will be executed after the <Configured transaction duration> is reached. - This causes the transaction to stop. 	
Tool validations	<p>N/a</p> <p>Post scenario validations:</p> <ul style="list-style-type: none"> - The TransactionEventRequest containing eventType <i>Ended</i> contains the MeterValue field. - The MeterValue must contain <An element per data collection moment indicated by SampledDataTxEndedInterval. The OCTT will not validate this.> - timestamp <The intervals between the timestamps of the received Meter Value messages must equal the configured value at SampledDataTxEndedInterval.> - sampledValue[0].context must be <i>Sample.Periodic</i> AND one must have <i>Transaction.End</i> - sampledValue must contain <An element per configured measurand at the SampledDataTxEndedMeasurands. The measurand field may be omitted when the measurand is "Energy.Active.Import.Register"> - sampledValue.signedMeterValue should not be omitted - sampledValue.signedMeterValue.publicKey should exist and depending on the value of OCPPCommCtrlr.PublicKeyWithSignedMeterValue, should be either "", or a valid public key 	

2.12. K SmartCharging

Table 189. Test Case Id: TC_K_38_CS

Test case name	Remote start transaction with charging profile - Ignore chargingProfile	
Test case Id	TC_K_38_CS	
Use case Id(s)	F01	
Requirement(s)	F01.FR.12	
System under test	Charging Station	
Description	The CSMS sets a TxProfile on a specific EVSE inside a RequestStartTransactionRequest message.	
Purpose	To verify if the Charging Station is able to ignore a TxProfile on a specific EVSE when receiving one in a RequestStartTransactionRequest message, when it does not support Smart Charging.	
Prerequisite(s)	The Charging Station does NOT support Smart Charging.	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): State is <i>EVConnectedPreSession</i>	
Main (Test scenario)	Charging Station 2. The Charging Station responds with a RequestStartTransactionResponse	CSMS 1. The OCTT sends a RequestStartTransactionRequest with idToken.idToken <Configured valid_idtoken_idtoken> idToken.type <Configured valid_idtoken_type> evsId <Configured evsId> chargingProfile.chargingProfilePurpose is <i>TxProfile</i> chargingProfile.transactionId is omitted. chargingProfile.chargingProfileKind is <i>Relative</i> chargingProfile.chargingSchedule.chargingSchedulePeriod[0].numberPhases <Configured numberPhases> chargingProfile.chargingSchedule.chargingSchedulePeriod[0].startPeriod 0 If <Configured chargingRateUnit> is A: chargingProfile.chargingSchedule.chargingSchedulePeriod[0].limit 6 If <Configured chargingRateUnit> is W: chargingProfile.chargingSchedule.chargingSchedulePeriod[0].limit 6000
	3. The Charging Station sends an AuthorizeRequest <u>Note(s):</u> - This step needs to be executed when AuthCtrlr.AuthorizeRemoteStart is true, unless (AuthEnabled is implemented with mutability <i>ReadOnly</i> AND the value is set to false) OR the idToken is cached. In case the idToken is used for a reservation, sending the AuthorizeRequest message is optional.	4. The OCTT responds with an AuthorizeResponse with idTokenInfo.status Accepted

Test case name	Remote start transaction with charging profile - Ignore chargingProfile	
	<p>5. The Charging Station sends a TransactionEventRequest</p> <p>Note(s):</p> <ul style="list-style-type: none"> - This step needs to be executed when TxStartPoint contains Authorized OR the transaction already started. So in the case TxStartPoint contains ParkingBayOccupancy or (EVConnected, in the case this testcase was initiated from state EVConnectedPreSession.) 	<p>6. The OCTT responds with a TransactionEventResponse</p> <p>Note(s):</p> <ul style="list-style-type: none"> - The first TransactionEventRequest sent after authorization contains the idToken field. The TransactionEventResponse of this request message contains idTokenInfo with status Accepted
	7. Execute Reusable State <i>EnergyTransferStarted</i>	
Tool validations	<ul style="list-style-type: none"> * Step 2: <p>Message: RequestStartTransactionResponse</p> <ul style="list-style-type: none"> - status must be Accepted <p>If the transaction has already been started, so if TxStartPoint contains ParkingBayOccupancy OR (TxStartPoint contains EVConnected AND State pre reusable state execution was EVConnectedPreSession) then</p> <ul style="list-style-type: none"> - transactionId must be <Provided transactionId in first TransactionEventRequest> <ul style="list-style-type: none"> * Step 3: <p>Message: AuthorizeRequest</p> <ul style="list-style-type: none"> - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type> <ul style="list-style-type: none"> * Step 5: <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - triggerReason must be RemoteStart - transactionInfo.remoteStartId must be present. <p>Post scenario validations: N/a</p>	

2.13. L Firmware Management

Table 190. Test Case Id: TC_L_01_CS

Test case name	Secure Firmware Update - Installation successful	
Test case Id	TC_L_01_CS	
Use case Id(s)	L01	
Requirement(s)	L01.FR.01,L01.FR.04,L01.FR.05,L01.FR.09,L01.FR.10,L01.FR.12,L01.FR.13,L01.FR.15,L01.FR.20,L01.FR.21,L01.FR.23	
System under test	Charging Station	
Description	The CSMS is able to request the Charging Station to securely download and install a new firmware by sending an UpdateFirmwareRequest with a signingCertificate .	
Purpose	To verify if the Charging Station is able to securely download and install a new firmware.	
Prerequisite(s)	A file server has been setup according to the (by the Charging Station) supported file transfer protocol(s), indicated by the configuration variable FileTransferProtocols .	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with a UpdateFirmwareResponse	1. The OCTT sends a UpdateFirmwareRequest with firmware.installDateTime <Current DateTime - 2 hours> firmware.location <Configured firmware_location> firmware.retrieveDateTime <Current DateTime - 2 hours> firmware.signingCertificate <Configured signingCertificate> firmware.signature <Configured signature>
	3. The Charging Station sends a FirmwareStatusNotificationRequest .	4. The OCTT responds with a FirmwareStatusNotificationResponse .
	5. The Charging Station sends a FirmwareStatusNotificationRequest .	6. The OCTT responds with a FirmwareStatusNotificationResponse .
	7. The Charging Station sends a FirmwareStatusNotificationRequest .	8. The OCTT responds with a FirmwareStatusNotificationResponse .
	9. The Charging Station sends a FirmwareStatusNotificationRequest .	10. The OCTT responds with a FirmwareStatusNotificationResponse .
	<u>Note(s):</u> - This step is optional if the Charging Station needs to reboot during a firmware update AND the bootloader is unable to send OCPP messages. - If the Charging Station does NOT need to reboot during a firmware update then this step needs to be executed and then proceed to step 17.	

Test case name	Secure Firmware Update - Installation successful	
	<p>11. The Charging Station sends a FirmwareStatusNotificationRequest.</p> <p>Note(s): - This step is optional if the Charging Station needs to reboot during a firmware update AND the bootloader is unable to send OCPP messages.</p>	12. The OCTT responds with a FirmwareStatusNotificationResponse .
	<p>13. The Charging Station sends a BootNotificationRequest</p>	<p>14. The OCTT responds with a BootNotificationResponse with status Accepted</p>
	<p>15. The Charging Station sends a SecurityEventNotificationRequest.</p>	<p>16. The OCTT responds with a SecurityEventNotificationResponse.</p>
	<p>17. The Charging Station notifies the CSMS about the current state of all connectors.</p>	<p>18. The OCTT responds accordingly.</p>
	<p>19. The Charging Station sends a FirmwareStatusNotificationRequest.</p>	<p>20. The OCTT responds with a FirmwareStatusNotificationResponse.</p>

Test case name	Secure Firmware Update - Installation successful
Tool validations	<ul style="list-style-type: none"> * Step 2: Message UpdateFirmwareResponse - status Accepted * Step 3: Message FirmwareStatusNotificationRequest - status Downloading * Step 5: Message FirmwareStatusNotificationRequest - status Downloaded * Step 7: Message FirmwareStatusNotificationRequest - status SignatureVerified * Step 9: Message FirmwareStatusNotificationRequest - status Installing * Step 11: Message FirmwareStatusNotificationRequest - status InstallRebooting * Step 13: Message BootNotificationRequest - reason FirmwareUpdate * Step 15: Message SecurityEventNotificationRequest - type FirmwareUpdated * Step 17: Message: StatusNotificationRequest - connectorStatus Available Message: NotifyEventRequest - eventData[0].trigger Delta - eventData[0].actualValue "Available" - eventData[0].component.name "Connector" - eventData[0].variable.name "AvailabilityState" * Step 19: Message FirmwareStatusNotificationRequest - status Installed
	Post scenario validations: N/a

Table 191. Test Case Id: TC_L_02_CS

Test case name	Secure Firmware Update - InstallScheduled	
Test case Id	TC_L_02_CS	
Use case Id(s)	L01	
Requirement(s)	L01.FR.01,L01.FR.04,L01.FR.05,L01.FR.09,L01.FR.10,L01.FR.12,L01.FR.15,L01.FR.16,L01.FR.20,L01.FR.21,L01.FR.23	
System under test	Charging Station	
Description	The CSMS is able to request the Charging Station to securely download and install a new firmware by sending an UpdateFirmwareRequest with a signingCertificate .	
Purpose	To verify if the Charging Station is able securely download a new firmware and schedule its installation.	
Prerequisite(s)	<ul style="list-style-type: none"> - A file server has been setup according to the (by the Charging Station) supported file transfer protocol(s), indicated by the configuration variable FileTransferProtocols. - The OCTT configuration firmware installDateTime needs to set to a future dateTime. 	
Before (Preparations)	<p>Configuration State: N/a</p> <p>Memory State: N/a</p> <p>Reusable State(s): N/a</p>	
Main (Test scenario)	<p>Charging Station</p> <p>2. The Charging Station responds with a UpdateFirmwareResponse</p> <p>3. The Charging Station sends a FirmwareStatusNotificationRequest.</p> <p>5. The Charging Station sends a FirmwareStatusNotificationRequest.</p> <p>7. The Charging Station sends a FirmwareStatusNotificationRequest.</p> <p>9. The Charging Station sends a FirmwareStatusNotificationRequest.</p>	<p>CSMS</p> <p>1. The OCTT sends a UpdateFirmwareRequest with firmware.location <Configured firmware_location> firmware.retrieveDateTime <Current DateTime - 2 hours> firmware.signingCertificate <Configured signingCertificate> firmware.signature <Configured signature> firmware.installDateTime <Current DateTime + <Configured Install Offset Period>></p> <p>4. The OCTT responds with a FirmwareStatusNotificationResponse.</p> <p>6. The OCTT responds with a FirmwareStatusNotificationResponse.</p> <p>8. The OCTT responds with a FirmwareStatusNotificationResponse.</p> <p>10. The OCTT responds with a FirmwareStatusNotificationResponse.</p>

Test case name	Secure Firmware Update - InstallScheduled	
	<p>11. The Charging Station sends a FirmwareStatusNotificationRequest.</p> <p><u>Note(s):</u></p> <ul style="list-style-type: none"> - This step needs to be executed after the configured <i>installDateTime</i> is reached. - This step is optional if the Charging Station needs to reboot during a firmware update AND the bootloader is unable to send OCPP messages. - If the Charging Station does NOT need to reboot during a firmware update then this step needs to be executed and then proceed to step 19. 	<p>12. The OCTT responds with a FirmwareStatusNotificationResponse.</p>
	<p>13. The Charging Station sends a FirmwareStatusNotificationRequest.</p> <p><u>Note(s):</u></p> <ul style="list-style-type: none"> - This step is optional if the Charging Station needs to reboot during a firmware update AND the bootloader is unable to send OCPP messages. 	<p>14. The OCTT responds with a FirmwareStatusNotificationResponse.</p>
	<p>15. The Charging Station sends a BootNotificationRequest</p>	<p>16. The OCTT responds with a BootNotificationResponse with status Accepted</p>
	<p>17. The Charging Station sends a SecurityEventNotificationRequest.</p>	<p>18. The OCTT responds with a SecurityEventNotificationResponse.</p>
	<p>19. The Charging Station notifies the CSMS about the current state of all connectors.</p>	<p>20. The OCTT responds accordingly.</p>
	<p>21. The Charging Station sends a FirmwareStatusNotificationRequest.</p>	<p>22. The OCTT responds with a FirmwareStatusNotificationResponse.</p>

Test case name	Secure Firmware Update - InstallScheduled
Tool validations	<ul style="list-style-type: none"> * Step 2: Message UpdateFirmwareResponse - status Accepted * Step 3: Message FirmwareStatusNotificationRequest - status Downloading * Step 5: Message FirmwareStatusNotificationRequest - status Downloaded * Step 7: Message FirmwareStatusNotificationRequest - status SignatureVerified * Step 9: Message FirmwareStatusNotificationRequest - status InstallScheduled * Step 11: Message FirmwareStatusNotificationRequest - status Installing * Step 13: Message FirmwareStatusNotificationRequest - status InstallRebooting * Step 15: Message BootNotificationRequest - reason FirmwareUpdate * Step 17: Message SecurityEventNotificationRequest - type FirmwareUpdated * Step 19: Message: StatusNotificationRequest - connectorStatus Available Message: NotifyEventRequest - eventData[0].trigger Delta - eventData[0].actualValue "Available" - eventData[0].component.name "Connector" - eventData[0].variable.name "AvailabilityState" * Step 21: Message FirmwareStatusNotificationRequest - status Installed
	Post scenario validations: N/a

Table 192. Test Case Id: TC_L_03_CS

Test case name	Secure Firmware Update - DownloadScheduled	
Test case Id	TC_L_03_CS	
Use case Id(s)	L01	
Requirement(s)	L01.FR.01,L01.FR.04,L01.FR.05,L01.FR.09,L01.FR.10,L01.FR.12,L01.FR.13,L01.FR.15,L01.FR.20,L01.FR.21,L01.FR.23	
System under test	Charging Station	
Description	The CSMS is able to request the Charging Station to securely download and install a new firmware by sending an UpdateFirmwareRequest with a signingCertificate .	
Purpose	To verify if the Charging Station is able to schedule securely downloading a new firmware.	
Prerequisite(s)	<ul style="list-style-type: none"> - A file server has been setup according to the (by the Charging Station) supported file transfer protocol(s), indicated by the configuration variable FileTransferProtocols. - The OCTT configuration firmware retrieveDateTime needs to set to a future dateTime. 	
Before (Preparations)	<p>Configuration State: N/a</p> <p>Memory State: N/a</p> <p>Reusable State(s): N/a</p>	
Main (Test scenario)	<p>Charging Station</p> <p>2. The Charging Station responds with a UpdateFirmwareResponse</p> <p>Note(s): - This step needs to be executed after the configured retrieveDateTime is reached.</p> <p>7. The Charging Station sends a FirmwareStatusNotificationRequest.</p> <p>9. The Charging Station sends a FirmwareStatusNotificationRequest.</p>	<p>CSMS</p> <p>1. The OCTT sends a UpdateFirmwareRequest with firmware.installDateTime <Current DateTime - 2 hours> firmware.location <Configured firmware_location> firmware.retrieveDateTime <Current DateTime + <Configured Download Offset Period>> firmware.signingCertificate <Configured signingCertificate> firmware.signature <Configured signature></p> <p>3. The Charging Station sends a FirmwareStatusNotificationRequest.</p> <p>5. The Charging Station sends a FirmwareStatusNotificationRequest.</p> <p>7. The Charging Station sends a FirmwareStatusNotificationRequest.</p> <p>9. The Charging Station sends a FirmwareStatusNotificationRequest.</p> <p>4. The OCTT responds with a FirmwareStatusNotificationResponse.</p> <p>6. The OCTT responds with a FirmwareStatusNotificationResponse.</p> <p>8. The OCTT responds with a FirmwareStatusNotificationResponse.</p> <p>10. The OCTT responds with a FirmwareStatusNotificationResponse.</p>

Test case name	Secure Firmware Update - DownloadScheduled	
	<p>11. The Charging Station sends a FirmwareStatusNotificationRequest.</p> <p><u>Note(s):</u></p> <ul style="list-style-type: none"> - This step needs to be executed after the configured <i>installDateTime</i> is reached. - This step is optional if the Charging Station needs to reboot during a firmware update AND the bootloader is unable to send OCPP messages. - If the Charging Station does NOT need to reboot during a firmware update then this step needs to be executed and then proceed to step 19. 	<p>12. The OCTT responds with a FirmwareStatusNotificationResponse.</p>
	<p>13. The Charging Station sends a FirmwareStatusNotificationRequest.</p> <p><u>Note(s):</u></p> <ul style="list-style-type: none"> - This step is optional if the Charging Station needs to reboot during a firmware update AND the bootloader is unable to send OCPP messages. 	<p>14. The OCTT responds with a FirmwareStatusNotificationResponse.</p>
	<p>15. The Charging Station sends a BootNotificationRequest</p>	<p>16. The OCTT responds with a BootNotificationResponse with status <i>Accepted</i></p>
	<p>17. The Charging Station sends a SecurityEventNotificationRequest</p>	<p>18. The OCTT responds with a SecurityEventNotificationResponse with type <i>FirmwareUpdated</i></p>
	<p>19. The Charging Station notifies the CSMS about the current state of all connectors.</p>	<p>20. The OCTT responds accordingly.</p>
	<p>21. The Charging Station sends a FirmwareStatusNotificationRequest.</p>	<p>22. The OCTT responds with a FirmwareStatusNotificationResponse.</p>

Test case name	Secure Firmware Update - DownloadScheduled
Tool validations	<ul style="list-style-type: none"> * Step 2: Message UpdateFirmwareResponse - status Accepted * Step 3: Message FirmwareStatusNotificationRequest - status DownloadScheduled * Step 5: Message FirmwareStatusNotificationRequest - status Downloading * Step 7: Message FirmwareStatusNotificationRequest - status Downloaded * Step 9: Message FirmwareStatusNotificationRequest - status SignatureVerified * Step 11: Message FirmwareStatusNotificationRequest - status Installing * Step 13: Message FirmwareStatusNotificationRequest - status InstallRebooting * Step 15: Message BootNotificationRequest - reason FirmwareUpdate * Step 17: Message SecurityEventNotificationRequest - type FirmwareUpdated * Step 19: Message: StatusNotificationRequest - connectorStatus Available Message: NotifyEventRequest - eventData[0].trigger Delta - eventData[0].actualValue "Available" - eventData[0].component.name "Connector" - eventData[0].variable.name "AvailabilityState" * Step 21: Message FirmwareStatusNotificationRequest - status Installed
	Post scenario validations: N/a

Table 193. Test Case Id: TC_L_05_CS

Test case name	Secure Firmware Update - InvalidCertificate	
Test case Id	TC_L_05_CS	
Use case Id(s)	L01	
Requirement(s)	L01.FR.01,L01.FR.02,L01.FR.10,L01.FR.20,L01.FR.21,L01.FR.22	
System under test	Charging Station	
Description	The CSMS is able to request the Charging Station to securely download and install a new firmware by sending an UpdateFirmwareRequest with a signingCertificate.	
Purpose	To verify if the Charging Station is able to identify it receiving an invalid signing certificate and report this to the CSMS.	
Prerequisite(s)	N/a	
Before (Preparations)	<p>Configuration State: * <Configured Invalid Firmware SigningCertificate> should be a trusted certificate and not be the same as the <Configured Valid Firmware SigningCertificate></p> <p>Memory State: N/a</p> <p>Reusable State(s): N/a</p>	
Main (Test scenario)	<p>Charging Station</p> <p>2. The Charging Station responds with a UpdateFirmwareResponse</p>	<p>CSMS</p> <p>1. The OCTT sends a UpdateFirmwareRequest with firmware.installDateTime <Current DateTime - 2 hours> firmware.location <Configured firmware_location> firmware.retrieveDateTime <Current DateTime - 2 hours> firmware.signingCertificate <Configured invalid firmware signingCertificate> firmware.signature <Configured signature></p>
Tool validations	<p>* Step 2: Message UpdateFirmwareResponse - status <i>InvalidCertificate OR RevokedCertificate</i></p> <p>Post scenario validations: N/a</p>	

Table 194. Test Case Id: TC_L_06_CS

Test case name	Secure Firmware Update - InvalidSignature	
Test case Id	TC_L_06_CS	
Use case Id(s)	L01	
Requirement(s)	L01.FR.01,L01.FR.03,L01.FR.04,L01.FR.10,L01.FR.20	
System under test	Charging Station	
Description	The CSMS is able to request the Charging Station to securely download and install a new firmware by sending an UpdateFirmwareRequest with a signingCertificate .	
Purpose	To verify if the Charging Station is able to identify if the signature is invalid and report this to the CSMS.	
Prerequisite(s)	A file server has been setup according to the (by the Charging Station) supported file transfer protocol(s), indicated by the configuration variable FileTransferProtocols .	
Before (Preparations)	Configuration State: <Configured invalid firmware signature> should be a real signature Memory State: N/a Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with a UpdateFirmwareResponse	1. The OCTT sends a UpdateFirmwareRequest with firmware.installDateTime <Current DateTime - 2 hours> firmware.location <Configured firmware_location> firmware.retrieveDateTime <Current DateTime - 2 hours> firmware.signingCertificate <Configured signingCertificate> firmware.signature <Configured invalid firmware signature>
	3. The Charging Station sends a FirmwareStatusNotificationRequest .	4. The OCTT responds with a FirmwareStatusNotificationResponse .
	5. The Charging Station sends a FirmwareStatusNotificationRequest .	6. The OCTT responds with a FirmwareStatusNotificationResponse .
	7. The Charging Station sends a FirmwareStatusNotificationRequest .	8. The OCTT responds with a FirmwareStatusNotificationResponse .
	9. The Charging Station sends a SecurityEventNotificationRequest .	10. The OCTT responds with a SecurityEventNotificationResponse .

Test case name	Secure Firmware Update - InvalidSignature
Tool validations	<ul style="list-style-type: none"> * Step 2: Message UpdateFirmwareResponse - status Accepted * Step 3: Message FirmwareStatusNotificationRequest - status Downloading * Step 5: Message FirmwareStatusNotificationRequest - status Downloaded * Step 7: Message FirmwareStatusNotificationRequest - status InvalidSignature * Step 9: Message SecurityEventNotificationRequest - type InvalidFirmwareSignature <p>Post scenario validations: N/a</p>

Table 195. Test Case Id: TC_L_07_CS

Test case name	Secure Firmware Update - DownloadFailed	
Test case Id	TC_L_07_CS	
Use case Id(s)	L01	
Requirement(s)	L01.FR.01,L01.FR.10,L01.FR.20	
System under test	Charging Station	
Description	The CSMS is able to request the Charging Station to securely download and install a new firmware by sending an UpdateFirmwareRequest with a signingCertificate .	
Purpose	To verify if the Charging Station is able to report to the CSMS when it is unable to download the new firmware.	
Prerequisite(s)	<ul style="list-style-type: none"> - A file server has been setup according to the (by the Charging Station) supported file transfer protocol(s), indicated by the configuration variable FileTransferProtocols. - The at the OCTT configured invalid firmware location needs to point to a not existing firmware file name. 	
Before (Preparations)	<p>Configuration State: N/a</p> <p>Memory State: N/a</p> <p>Reusable State(s): N/a</p>	
Main (Test scenario)	<p>Charging Station</p> <p>2. The Charging Station responds with a UpdateFirmwareResponse</p> <p>Note(s): - This step is optional. The Charging Station may immediately identify downloading the firmware is not possible. </p> <p>3. The Charging Station sends a FirmwareStatusNotificationRequest.</p> <p>5. The Charging Station sends a FirmwareStatusNotificationRequest.</p> <p>Post scenario validations: N/a</p>	<p>CSMS</p> <p>1. The OCTT sends a UpdateFirmwareRequest with firmware.installDateTime <Current DateTime - 2 hours> firmware.location <Configured firmware location> + "_does_not_exist" firmware.retrieveDateTime <Current DateTime - 2 hours> firmware.signingCertificate <Configured signingCertificate> firmware.signature <Configured signature></p> <p>4. The OCTT responds with a FirmwareStatusNotificationResponse.</p> <p>6. The OCTT responds with a FirmwareStatusNotificationResponse.</p>
Tool validations	<ul style="list-style-type: none"> * Step 2: Message UpdateFirmwareResponse - status Accepted * Step 3: Message FirmwareStatusNotificationRequest - status Downloading * Step 5: Message FirmwareStatusNotificationRequest - status DownloadFailed 	

Table 196. Test Case Id: TC_L_08_CS

Test case name	Secure Firmware Update - InstallVerificationFailed or InstallationFailed	
Test case Id	TC_L_08_CS	
Use case Id(s)	L01	
Requirement(s)	L01.FR.01,L01.FR.10,L01.FR.12,L01.FR.20	
System under test	Charging Station	
Description	The CSMS is able to request the Charging Station to securely download and install a new firmware by sending an UpdateFirmwareRequest with a signingCertificate.	
Purpose	To verify if the Charging Station is able to report to the CSMS when the firmware verification fails.	
Prerequisite(s)	<ul style="list-style-type: none"> - A file server has been setup according to the (by the Charging Station) supported file transfer protocol(s), indicated by the configuration variable FileTransferProtocols. - The at the OCTT configured invalid firmware location needs to point to a firmware file that causes an InstallVerificationFailed. 	
Before (Preparations)	<p>Configuration State:</p> <ul style="list-style-type: none"> * <Configured invalid firmware location> should point to existing firmware that causes an InstallVerificationFailed * <Configured invalid firmware signingCertificate> should be a trusted signingCertificate * <Configured invalid firmware signature> should be a real signature <p>Memory State: N/a</p> <p>Reusable State(s): N/a</p>	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with a UpdateFirmwareResponse	1. The OCTT sends a UpdateFirmwareRequest with firmware.installDateTime <Current DateTime - 2 hours> firmware.location <Configured invalid firmware location> firmware.retrieveDateTime <Current DateTime + <Configured Download Offset Period>> firmware.signingCertificate <Configured invalid firmware signingCertificate> firmware.signature <Configured invalid firmware signature>
	3. The Charging Station sends a FirmwareStatusNotificationRequest .	4. The OCTT responds with a FirmwareStatusNotificationResponse .
	5. The Charging Station sends a FirmwareStatusNotificationRequest .	6. The OCTT responds with a FirmwareStatusNotificationResponse .
	Note(s): - This step needs to be executed after the configured retrieveDateTime is reached.	
	7. The Charging Station sends a FirmwareStatusNotificationRequest .	8. The OCTT responds with a FirmwareStatusNotificationResponse .
	9. The Charging Station sends a FirmwareStatusNotificationRequest .	10. The OCTT responds with a FirmwareStatusNotificationResponse .

Test case name	Secure Firmware Update - InstallVerificationFailed or InstallationFailed	
	11. The Charging Station sends a FirmwareStatusNotificationRequest .	12. The OCTT responds with a FirmwareStatusNotificationResponse .
	13. The Charging Station sends a FirmwareStatusNotificationRequest .	14. The OCTT responds with a FirmwareStatusNotificationResponse .
	15. The Charging Station sends a BootNotificationRequest	16. The OCTT responds with a BootNotificationResponse with status Accepted
	17. The Charging Station notifies the CSMS about the current state of all connectors.	18. The OCTT responds accordingly.
	19. The Charging Station sends a FirmwareStatusNotificationRequest .	20. The OCTT responds with a FirmwareStatusNotificationResponse .
<p><u>Note(s):</u></p> <p>- Steps 13, 14, 15, 16, 17, 18 are optional if the Charging Station needs to reboot during a firmware update AND the bootloader is unable to send OCPP messages.</p>		
Tool validations	<ul style="list-style-type: none"> * Step 2: Message UpdateFirmwareResponse - status Accepted * Step 3: Message FirmwareStatusNotificationRequest - status DownloadScheduled * Step 5: Message FirmwareStatusNotificationRequest - status Downloading * Step 7: Message FirmwareStatusNotificationRequest - status Downloaded * Step 9: Message FirmwareStatusNotificationRequest - status SignatureVerified * Step 11: Message FirmwareStatusNotificationRequest - status Installing * Step 13: Message FirmwareStatusNotificationRequest - status InstallRebooting * Step 15: Message BootNotificationRequest - reason FirmwareUpdate * Step 17: Message: StatusNotificationRequest - connectorStatus Available Message: NotifyEventRequest - eventData[0].trigger Delta - eventData[0].actualValue "Available" - eventData[0].component.name "Connector" - eventData[0].variable.name "AvailabilityState" * Step 19: Message FirmwareStatusNotificationRequest - status InstallVerificationFailed or InstallationFailed <p>Post scenario validations: N/a</p>	

Table 197. Test Case Id: TC_L_10_CS

Test case name	Secure Firmware Update - AcceptedCanceled	
Test case Id	TC_L_10_CS	
Use case Id(s)	L01	
Requirement(s)	L01.FR.01,L01.FR.10,L01.FR.20,L01.FR.24	
System under test	Charging Station	
Description	The CSMS is able to request the Charging Station to securely download and install a new firmware by sending an UpdateFirmwareRequest with a signingCertificate.	
Purpose	To verify if the Charging Station is able to cancel an ongoing firmware update and start a new one, when receiving an UpdateFirmwareRequest from the CSMS.	
Prerequisite(s)	<ul style="list-style-type: none"> - A file server has been setup according to the (by the Charging Station) supported file transfer protocol(s), indicated by the configuration variable FileTransferProtocols. - The Charging Station is able to cancel an ongoing firmware update while it is busy downloading a new firmware file. 	
Before (Preparations)	<p>Configuration State: N/a</p> <p>Memory State: N/a</p> <p>Reusable State(s): N/a</p>	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with a UpdateFirmwareResponse	<p>1. The OCTT sends a UpdateFirmwareRequest with firmware.installDateTime <Current DateTime - 2 hours></p> <p>firmware.location <Configured firmware_location></p> <p>firmware.retrieveDateTime <Current DateTime - 2 hours></p> <p>firmware.signingCertificate <Configured signingCertificate></p> <p>firmware.signature <Configured signature></p>
	3. The Charging Station sends a FirmwareStatusNotificationRequest .	<p>4. The OCTT responds with a FirmwareStatusNotificationResponse.</p>
	6. The Charging Station responds with a UpdateFirmwareResponse	<p>5. The OCTT sends a UpdateFirmwareRequest with firmware.installDateTime <Current DateTime - 2 hours></p> <p>firmware.location <Configured firmware_location></p> <p>firmware.retrieveDateTime <Current DateTime - 2 hours></p> <p>firmware.signingCertificate <Configured signingCertificate></p> <p>firmware.signature <Configured signature></p>
	7. The Charging Station sends a FirmwareStatusNotificationRequest .	<p>8. The OCTT responds with a FirmwareStatusNotificationResponse.</p>
	9. The Charging Station sends a FirmwareStatusNotificationRequest .	<p>10. The OCTT responds with a FirmwareStatusNotificationResponse.</p>

Test case name	Secure Firmware Update - AcceptedCanceled	
	<p>11. The Charging Station sends a FirmwareStatusNotificationRequest.</p>	<p>12. The OCTT responds with a FirmwareStatusNotificationResponse.</p>
	<p>13. The Charging Station sends a FirmwareStatusNotificationRequest.</p> <p><u>Note(s):</u> - This step is optional if the Charging Station needs to reboot during a firmware update AND the bootloader is unable to send OCPP messages. - If the Charging Station does NOT need to reboot during a firmware update then this step needs to be executed and then proceed to step 21.</p>	<p>14. The OCTT responds with a FirmwareStatusNotificationResponse.</p>
	<p>15. The Charging Station sends a FirmwareStatusNotificationRequest.</p> <p><u>Note(s):</u> - This step is optional if the Charging Station needs to reboot during a firmware update AND the bootloader is unable to send OCPP messages.</p>	<p>16. The OCTT responds with a FirmwareStatusNotificationResponse.</p>
	<p>17. The Charging Station sends a BootNotificationRequest</p>	<p>18. The OCTT responds with a BootNotificationResponse with status Accepted</p>
	<p>19. The Charging Station sends a SecurityEventNotificationRequest w with type FirmwareUpdated</p>	<p>20. The OCTT responds with a SecurityEventNotificationResponse</p>
	<p>21. The Charging Station notifies the CSMS about the current state of all connectors.</p>	<p>22. The OCTT responds accordingly.</p>
	<p>23. The Charging Station sends a FirmwareStatusNotificationRequest.</p>	<p>24. The OCTT responds with a FirmwareStatusNotificationResponse.</p>

Test case name	Secure Firmware Update - AcceptedCanceled
Tool validations	<ul style="list-style-type: none"> * Step 2: Message UpdateFirmwareResponse - status Accepted * Step 3: Message FirmwareStatusNotificationRequest - status Downloading * Step 6: Message UpdateFirmwareResponse - status AcceptedCanceled * Step 7: Message FirmwareStatusNotificationRequest - status Downloading * Step 9: Message FirmwareStatusNotificationRequest - status Downloaded * Step 11: Message FirmwareStatusNotificationRequest - status SignatureVerified * Step 13: Message FirmwareStatusNotificationRequest - status Installing * Step 15: Message FirmwareStatusNotificationRequest - status InstallRebooting * Step 17: Message BootNotificationRequest - reason FirmwareUpdate * Step 19: Message SecurityEventNotificationRequest - type FirmwareUpdated * Step 21: Message: StatusNotificationRequest - connectorStatus Available Message: NotifyEventRequest - eventData[0].trigger Delta - eventData[0].actualValue "Available" - eventData[0].component.name "Connector" - eventData[0].variable.name "AvailabilityState" * Step 23: Message FirmwareStatusNotificationRequest - status Installed
	Post scenario validations: N/a

Table 198. Test Case Id: TC_L_11_CS

Test case name	Secure Firmware Update - Unable to cancel	
Test case Id	TC_L_11_CS	
Use case Id(s)	L01	
Requirement(s)	L01.FR.01,L01.FR.10,L01.FR.20,L01.FR.27	
System under test	Charging Station	
Description	The CSMS is able to request the Charging Station to securely download and install a new firmware by sending an UpdateFirmwareRequest with a signingCertificate .	
Purpose	To verify if the Charging Station is able to reject a firmware update request when it is unable to cancel an ongoing firmware update.	
Prerequisite(s)	<ul style="list-style-type: none"> - A file server has been setup according to the (by the Charging Station) supported file transfer protocol(s), indicated by the configuration variable FileTransferProtocols. - The Charging Station is NOT able to cancel an ongoing firmware update. 	
Before (Preparations)	<p>Configuration State: N/a</p> <p>Memory State: N/a</p> <p>Reusable State(s): N/a</p>	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with a UpdateFirmwareResponse	<p>1. The OCTT sends a UpdateFirmwareRequest with firmware.installDateTime <Current DateTime - 2 hours></p> <p>firmware.location <Configured firmware_location></p> <p>firmware.retrieveDateTime <Current DateTime - 2 hours></p> <p>firmware.signingCertificate <Configured signingCertificate></p> <p>firmware.signature <Configured signature></p>
	3. The Charging Station sends a FirmwareStatusNotificationRequest .	<p>4. The OCTT responds with a FirmwareStatusNotificationResponse.</p>
	6. The Charging Station responds with a UpdateFirmwareResponse	<p>5. The OCTT sends a UpdateFirmwareRequest with firmware.location <Configured firmware_location></p> <p>firmware.retrieveDateTime <Current DateTime></p> <p>firmware.signingCertificate <Configured signingCertificate></p> <p>firmware.signature <Configured signature></p>
	7. The Charging Station sends a FirmwareStatusNotificationRequest .	<p>8. The OCTT responds with a FirmwareStatusNotificationResponse.</p>
	9. The Charging Station sends a FirmwareStatusNotificationRequest .	<p>10. The OCTT responds with a FirmwareStatusNotificationResponse.</p>

Test case name	Secure Firmware Update - Unable to cancel	
	<p>11. The Charging Station sends a FirmwareStatusNotificationRequest.</p> <p>Note(s): - This step is optional if the Charging Station needs to reboot during a firmware update AND the bootloader is unable to send OCPP messages. - If the Charging Station does NOT need to reboot during a firmware update then this step needs to be executed and then proceed to step 19.</p>	<p>12. The OCTT responds with a FirmwareStatusNotificationResponse.</p>
	<p>13. The Charging Station sends a FirmwareStatusNotificationRequest.</p> <p>Note(s): - This step is optional if the Charging Station needs to reboot during a firmware update AND the bootloader is unable to send OCPP messages.</p>	<p>14. The OCTT responds with a FirmwareStatusNotificationResponse.</p>
	<p>15. The Charging Station sends a BootNotificationRequest</p>	<p>16. The OCTT responds with a BootNotificationResponse with status Accepted</p>
	<p>17. The Charging Station sends a SecurityEventNotificationRequest with type FirmwareUpdated</p>	<p>18. The OCTT responds with a SecurityEventNotificationResponse</p>
	<p>19. The Charging Station notifies the CSMS about the current state of all connectors.</p>	<p>20. The OCTT responds accordingly.</p>
	<p>21. The Charging Station sends a FirmwareStatusNotificationRequest.</p>	<p>22. The OCTT responds with a FirmwareStatusNotificationResponse.</p>

Test case name	Secure Firmware Update - Unable to cancel
Tool validations	<ul style="list-style-type: none"> * Step 2: Message UpdateFirmwareResponse - status Accepted * Step 3: Message FirmwareStatusNotificationRequest - status Downloading * Step 6: Message UpdateFirmwareResponse - status Rejected * Step 7: Message FirmwareStatusNotificationRequest - status Downloaded * Step 9: Message FirmwareStatusNotificationRequest - status SignatureVerified * Step 11: Message FirmwareStatusNotificationRequest - status Installing * Step 13: Message FirmwareStatusNotificationRequest - status InstallRebooting * Step 15: Message BootNotificationRequest - reason FirmwareUpdate * Step 17: Message SecurityEventNotificationRequest - type FirmwareUpdated * Step 19: Message: StatusNotificationRequest - connectorStatus Available Message: NotifyEventRequest - eventData[0].trigger Delta - eventData[0].actualValue "Available" - eventData[0].component.name "Connector" - eventData[0].variable.name "AvailabilityState" * Step 21: Message FirmwareStatusNotificationRequest - status Installed
	Post scenario validations: N/a

Table 199. Test Case Id: TC_L_12_CS

Test case name	Secure Firmware Update - Unable to download/install firmware with ongoing transaction - AllowNewSessionsPendingFirmwareUpdate is true	
Test case Id	TC_L_12_CS	
Use case Id(s)	L01	
Requirement(s)	L01.FR.01,L01.FR.06,L01.FR.07,L01.FR.10,L01.FR.20	
System under test	Charging Station	
Description	The CSMS is able to request the Charging Station to securely download and install a new firmware by sending an UpdateFirmwareRequest with a signingCertificate.	
Purpose	To verify if the Charging Station is able to keep allowing new transactions when requested to update the firmware, while there is an ongoing transaction.	
Prerequisite(s)	<ul style="list-style-type: none"> - A file server has been setup according to the (by the Charging Station) supported file transfer protocol(s), indicated by the configuration variable FileTransferProtocols. - The Charging Station is able to start more than one transaction at a time. - The Charging Station is unable to download AND install firmware while there is an ongoing transaction. 	
Before (Preparations)	<p>Configuration State: AllowNewSessionsPendingFirmwareUpdate is true (If implemented)</p> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>EnergyTransferStarted</i> for <Configured connectorId></p>	
Main (Test scenario)	<p>Charging Station</p> <p>2. The Charging Station responds with a UpdateFirmwareResponse</p> <p>3. The Charging Station sends a FirmwareStatusNotificationRequest.</p> <p>5. Execute Reusable State <i>EnergyTransferStarted</i> for <Configured second Connector></p> <p>6. Execute Reusable State <i>ParkingBayUnoccupied</i> for <Configured connectorId></p> <p>Note(s): - This causes the transaction to stop.</p> <p>7. Execute Reusable State <i>ParkingBayUnoccupied</i> for <Configured second Connector></p> <p>Note(s): - This causes the transaction to stop.</p> <p>8. The Charging Station sends a FirmwareStatusNotificationRequest.</p> <p>10. The Charging Station sends a FirmwareStatusNotificationRequest.</p> <p>12. The Charging Station sends a FirmwareStatusNotificationRequest.</p>	<p>CSMS</p> <p>1. The OCTT sends a UpdateFirmwareRequest with firmware.installDateTime <Current DateTime - 2 hours> firmware.location <Configured firmware_location> firmware.retrieveDateTime <Current DateTime - 2 hours> firmware.signingCertificate <Configured signingCertificate> firmware.signature <Configured signature></p> <p>4. The OCTT responds with a FirmwareStatusNotificationResponse.</p> <p>9. The OCTT responds with a FirmwareStatusNotificationResponse.</p> <p>11. The OCTT responds with a FirmwareStatusNotificationResponse.</p> <p>13. The OCTT responds with a FirmwareStatusNotificationResponse.</p>

Test case name	Secure Firmware Update - Unable to download/install firmware with ongoing transaction - AllowNewSessionsPendingFirmwareUpdate is true	
	<p>14. The Charging Station sends a FirmwareStatusNotificationRequest.</p> <p><u>Note(s):</u></p> <ul style="list-style-type: none"> - This step is optional if the Charging Station needs to reboot during a firmware update AND the bootloader is unable to send OCPP messages. - If the Charging Station does NOT need to reboot during a firmware update then this step needs to be executed and then proceed to step 22. 	<p>15. The OCTT responds with a FirmwareStatusNotificationResponse.</p>
	<p>16. The Charging Station sends a FirmwareStatusNotificationRequest.</p> <p><u>Note(s):</u></p> <ul style="list-style-type: none"> - This step is optional if the Charging Station needs to reboot during a firmware update AND the bootloader is unable to send OCPP messages. 	<p>17. The OCTT responds with a FirmwareStatusNotificationResponse.</p>
	<p>18. The Charging Station sends a BootNotificationRequest</p>	<p>19. The OCTT responds with a BootNotificationResponse with status Accepted</p>
	<p>20. The Charging Station sends a SecurityEventNotificationRequest</p>	<p>21. The OCTT responds with a SecurityEventNotificationResponse</p>
	<p>22. The Charging Station notifies the CSMS about the current state of all connectors.</p>	<p>23. The OCTT responds accordingly.</p>
	<p>24. The Charging Station sends a FirmwareStatusNotificationRequest.</p>	<p>25. The OCTT responds with a FirmwareStatusNotificationResponse.</p>

Test case name	Secure Firmware Update - Unable to download/install firmware with ongoing transaction - AllowNewSessionsPendingFirmwareUpdate is true
Tool validations	<ul style="list-style-type: none"> * Step 2: Message UpdateFirmwareResponse - status Accepted * Step 3: Message FirmwareStatusNotificationRequest - status DownloadScheduled * Step 8: Message FirmwareStatusNotificationRequest - status Downloading * Step 10: Message FirmwareStatusNotificationRequest - status Downloaded * Step 12: Message FirmwareStatusNotificationRequest - status SignatureVerified * Step 14: Message FirmwareStatusNotificationRequest - status Installing * Step 16: Message FirmwareStatusNotificationRequest - status InstallRebooting * Step 18: Message BootNotificationRequest - reason FirmwareUpdate * Step 20: Message SecurityEventNotificationRequest - type FirmwareUpdated * Step 22: Message: StatusNotificationRequest - connectorStatus Available Message: NotifyEventRequest - eventData[0].trigger Delta - eventData[0].actualValue "Available" - eventData[0].component.name "Connector" - eventData[0].variable.name "AvailabilityState" * Step 24: Message FirmwareStatusNotificationRequest - status Installed
	Post scenario validations: N/a

Table 200. Test Case Id: TC_L_13_CS

Test case name	Secure Firmware Update - Unable to download/install firmware with ongoing transaction - AllowNewSessionsPendingFirmwareUpdate is false	
Test case Id	TC_L_13_CS	
Use case Id(s)	L01	
Requirement(s)	L01.FR.01,L01.FR.06,L01.FR.07,L01.FR.10,L01.FR.20	
System under test	Charging Station	
Description	The CSMS is able to request the Charging Station to securely download and install a new firmware by sending an UpdateFirmwareRequest with a signingCertificate.	
Purpose	To verify if the Charging Station is able to set its available connectors to Unavailable when requested to update the firmware, while there is an ongoing transaction.	
Prerequisite(s)	<ul style="list-style-type: none"> - A file server has been setup according to the (by the Charging Station) supported file transfer protocol(s), indicated by the configuration variable FileTransferProtocols. - The configuration variable AllowNewSessionsPendingFirmwareUpdate is implemented. - The Charging Station is unable to download AND install firmware while there is an ongoing transaction. 	
Before (Preparations)	<p>Configuration State: AllowNewSessionsPendingFirmwareUpdate is <i>false</i></p> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>EnergyTransferStarted</i></p>	
Main (Test scenario)	<p>Charging Station</p> <p>2. The Charging Station responds with a UpdateFirmwareResponse</p> <p>Note(s): - This step needs to be executed for all connectors with AvailabilityState Available.</p> <p>7. Execute Reusable State <i>ParkingBayUnoccupied</i></p> <p>Note(s): - This causes the transaction to stop.</p> <p>8. The Charging Station sends a FirmwareStatusNotificationRequest.</p> <p>10. The Charging Station sends a FirmwareStatusNotificationRequest.</p> <p>12. The Charging Station sends a FirmwareStatusNotificationRequest.</p>	<p>CSMS</p> <p>1. The OCTT sends a UpdateFirmwareRequest with firmware.installDateTime <Current DateTime - 2 hours> firmware.location <Configured firmware_location> firmware.retrieveDateTime <Current DateTime - 2 hours> firmware.signingCertificate <Configured signingCertificate> firmware.signature <Configured signature></p> <p>3. The Charging Station sends a FirmwareStatusNotificationRequest.</p> <p>5. The Charging Station notifies the CSMS about the current state of its Available connector(s).</p> <p>6. The OCTT responds accordingly.</p> <p>9. The OCTT responds with a FirmwareStatusNotificationResponse.</p> <p>11. The OCTT responds with a FirmwareStatusNotificationResponse.</p> <p>13. The OCTT responds with a FirmwareStatusNotificationResponse.</p>

Test case name	Secure Firmware Update - Unable to download/install firmware with ongoing transaction - AllowNewSessionsPendingFirmwareUpdate is false	
	<p>14. The Charging Station sends a FirmwareStatusNotificationRequest.</p> <p><u>Note(s):</u></p> <ul style="list-style-type: none"> - This step is optional if the Charging Station needs to reboot during a firmware update AND the bootloader is unable to send OCPP messages. - If the Charging Station does NOT need to reboot during a firmware update then this step needs to be executed and then proceed to step 22. 	<p>15. The OCTT responds with a FirmwareStatusNotificationResponse.</p>
	<p>16. The Charging Station sends a FirmwareStatusNotificationRequest.</p> <p><u>Note(s):</u></p> <ul style="list-style-type: none"> - This step is optional if the Charging Station needs to reboot during a firmware update AND the bootloader is unable to send OCPP messages. 	<p>17. The OCTT responds with a FirmwareStatusNotificationResponse.</p>
	<p>18. The Charging Station sends a BootNotificationRequest</p>	<p>19. The OCTT responds with a BootNotificationResponse with status Accepted</p>
	<p>20. The Charging Station sends a SecurityEventNotificationRequest</p>	<p>21. The OCTT responds with a SecurityEventNotificationResponse</p>
	<p>22. The Charging Station notifies the CSMS about the current state of all connectors.</p>	<p>23. The OCTT responds accordingly.</p>
	<p>24. The Charging Station sends a FirmwareStatusNotificationRequest.</p>	<p>25. The OCTT responds with a FirmwareStatusNotificationResponse.</p>

Test case name	Secure Firmware Update - Unable to download/install firmware with ongoing transaction - AllowNewSessionsPendingFirmwareUpdate is false
Tool validations	<ul style="list-style-type: none"> * Step 2: Message UpdateFirmwareResponse - status Accepted * Step 3: Message FirmwareStatusNotificationRequest - status DownloadScheduled * Step 5: Message: StatusNotificationRequest - connectorStatus Unavailable Message: NotifyEventRequest - eventData[0].trigger Delta - eventData[0].actualValue "Unavailable" - eventData[0].component.name "Connector" - eventData[0].variable.name "AvailabilityState" * Step 8: Message FirmwareStatusNotificationRequest - status Downloading * Step 10: Message FirmwareStatusNotificationRequest - status Downloaded * Step 12: Message FirmwareStatusNotificationRequest - status SignatureVerified * Step 14: Message FirmwareStatusNotificationRequest - status Installing * Step 16: Message FirmwareStatusNotificationRequest - status InstallRebooting
	<ul style="list-style-type: none"> * Step 18: Message BootNotificationRequest - reason FirmwareUpdate * Step 20: Message SecurityEventNotificationRequest - type FirmwareUpdated * Step 22: Message: StatusNotificationRequest - connectorStatus Available Message: NotifyEventRequest - eventData[0].trigger Delta - eventData[0].actualValue "Available" - eventData[0].component.name "Connector" - eventData[0].variable.name "AvailabilityState" * Step 24: Message FirmwareStatusNotificationRequest - status Installed <p>Post scenario validations: N/a</p>

Table 201. Test Case Id: TC_L_14_CS

Test case name	Secure Firmware Update - Unable to install firmware with ongoing transaction - AllowNewSessionsPendingFirmwareUpdate is true	
Test case Id	TC_L_14_CS	
Use case Id(s)	L01	
Requirement(s)	L01.FR.01,L01.FR.06,L01.FR.07,L01.FR.10,L01.FR.20	
System under test	Charging Station	
Description	The CSMS is able to request the Charging Station to securely download and install a new firmware by sending an UpdateFirmwareRequest with a signingCertificate.	
Purpose	To verify if the Charging Station is able to keep allowing new transactions when requested to update the firmware, while there is an ongoing transaction.	
Prerequisite(s)	<ul style="list-style-type: none"> - A file server has been setup according to the (by the Charging Station) supported file transfer protocol(s), indicated by the configuration variable FileTransferProtocols. - The Charging Station is able to start more than one transaction at a time. - The Charging Station is unable to install firmware while there is an ongoing transaction. 	
Before (Preparations)	<p>Configuration State: AllowNewSessionsPendingFirmwareUpdate is true (If implemented)</p> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>EnergyTransferStarted</i> for <Configured connectorId></p>	
Main (Test scenario)	<p>Charging Station</p> <p>2. The Charging Station responds with a UpdateFirmwareResponse</p> <p>3. The Charging Station sends a FirmwareStatusNotificationRequest.</p> <p>5. The Charging Station sends a FirmwareStatusNotificationRequest.</p> <p>7. The Charging Station sends a FirmwareStatusNotificationRequest.</p> <p>9. The Charging Station sends a FirmwareStatusNotificationRequest.</p> <p>11. Execute Reusable State <i>EnergyTransferStarted</i> for <Configured second Connector></p> <p>12. Execute Reusable State <i>ParkingBayUnoccupied</i> for <Configured connectorId></p> <p>Note(s): - This causes the transaction to stop.</p> <p>13. Execute Reusable State <i>ParkingBayUnoccupied</i> for <Configured second Connector></p> <p>Note(s): - This causes the transaction to stop.</p>	<p>CSMS</p> <p>1. The OCTT sends a UpdateFirmwareRequest with firmware.installDateTime <Current DateTime - 2 hours> firmware.location <Configured firmware_location> firmware.retrieveDateTime <Current DateTime - 2 hours> firmware.signingCertificate <Configured signingCertificate> firmware.signature <Configured signature></p> <p>4. The OCTT responds with a FirmwareStatusNotificationResponse.</p> <p>6. The OCTT responds with a FirmwareStatusNotificationResponse.</p> <p>8. The OCTT responds with a FirmwareStatusNotificationResponse.</p> <p>10. The OCTT responds with a FirmwareStatusNotificationResponse.</p>

Test case name	Secure Firmware Update - Unable to install firmware with ongoing transaction - AllowNewSessionsPendingFirmwareUpdate is true	
	<p>14. The Charging Station sends a FirmwareStatusNotificationRequest.</p> <p><u>Note(s):</u></p> <ul style="list-style-type: none"> - This step is optional if the Charging Station needs to reboot during a firmware update AND the bootloader is unable to send OCPP messages. - If the Charging Station does NOT need to reboot during a firmware update then this step needs to be executed and then proceed to step 22. 	<p>15. The OCTT responds with a FirmwareStatusNotificationResponse.</p>
	<p>16. The Charging Station sends a FirmwareStatusNotificationRequest.</p> <p><u>Note(s):</u></p> <ul style="list-style-type: none"> - This step is optional if the Charging Station needs to reboot during a firmware update AND the bootloader is unable to send OCPP messages. 	<p>17. The OCTT responds with a FirmwareStatusNotificationResponse.</p>
	<p>18. The Charging Station sends a BootNotificationRequest</p>	<p>19. The OCTT responds with a BootNotificationResponse with status Accepted</p>
	<p>20. The Charging Station sends a SecurityEventNotificationRequest with type FirmwareUpdated</p>	<p>21. The OCTT responds with a SecurityEventNotificationResponse</p>
	<p>22. The Charging Station notifies the CSMS about the current state of all connectors.</p>	<p>23. The OCTT responds accordingly.</p>
	<p>24. The Charging Station sends a FirmwareStatusNotificationRequest.</p>	<p>25. The OCTT responds with a FirmwareStatusNotificationResponse.</p>

Test case name	Secure Firmware Update - Unable to install firmware with ongoing transaction - AllowNewSessionsPendingFirmwareUpdate is true
Tool validations	<ul style="list-style-type: none"> * Step 2: Message UpdateFirmwareResponse - status Accepted * Step 3: Message FirmwareStatusNotificationRequest - status Downloading * Step 5: Message FirmwareStatusNotificationRequest - status Downloaded * Step 7: Message FirmwareStatusNotificationRequest - status SignatureVerified * Step 9: Message FirmwareStatusNotificationRequest - status InstallScheduled * Step 14: Message FirmwareStatusNotificationRequest - status Installing * Step 16: Message FirmwareStatusNotificationRequest - status InstallRebooting * Step 18: Message BootNotificationRequest - reason FirmwareUpdate * Step 20: Message SecurityEventNotificationRequest - type FirmwareUpdated * Step 22: Message: StatusNotificationRequest - connectorStatus Available Message: NotifyEventRequest - eventData[0].trigger Delta - eventData[0].actualValue "Available" - eventData[0].component.name "Connector" - eventData[0].variable.name "AvailabilityState" * Step 24: Message FirmwareStatusNotificationRequest - status Installed <p>Post scenario validations: N/a</p>

Table 202. Test Case Id: TC_L_15_CS

Test case name	Secure Firmware Update - Unable to install firmware with ongoing transaction - AllowNewSessionsPendingFirmwareUpdate is false	
Test case Id	TC_L_15_CS	
Use case Id(s)	L01	
Requirement(s)	L01.FR.01,L01.FR.06,L01.FR.07,L01.FR.10,L01.FR.20	
System under test	Charging Station	
Description	The CSMS is able to request the Charging Station to securely download and install a new firmware by sending an UpdateFirmwareRequest with a signingCertificate.	
Purpose	To verify if the Charging Station is able to set its available connectors to Unavailable when requested to update the firmware, while there is an ongoing transaction.	
Prerequisite(s)	<ul style="list-style-type: none"> - A file server has been setup according to the (by the Charging Station) supported file transfer protocol(s), indicated by the configuration variable FileTransferProtocols. - The configuration variable AllowNewSessionsPendingFirmwareUpdate is implemented. - The Charging Station is unable to install firmware while there is an ongoing transaction. 	
Before (Preparations)	<p>Configuration State: AllowNewSessionsPendingFirmwareUpdate is <i>false</i></p> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>EnergyTransferStarted</i></p>	
Main (Test scenario)	<p>Charging Station</p> <p>2. The Charging Station responds with a UpdateFirmwareResponse</p> <p>Note(s): - This step needs to be executed for all connectors with AvailabilityState Available.</p> <p>3. The Charging Station notifies the CSMS about the current state of its Available connector(s).</p> <p>5. The Charging Station sends a FirmwareStatusNotificationRequest.</p> <p>7. The Charging Station sends a FirmwareStatusNotificationRequest.</p> <p>9. The Charging Station sends a FirmwareStatusNotificationRequest.</p> <p>11. The Charging Station sends a FirmwareStatusNotificationRequest.</p> <p>13. Execute Reusable State <i>ParkingBayUnoccupied</i></p> <p>Note(s): - This causes the transaction to stop.</p>	<p>CSMS</p> <p>1. The OCTT sends a UpdateFirmwareRequest with firmware.installDateTime <Current DateTime - 2 hours> firmware.location <Configured firmware_location> firmware.retrieveDateTime <Current DateTime - 2 hours> firmware.signingCertificate <Configured signingCertificate> firmware.signature <Configured signature></p> <p>4. The OCTT responds accordingly.</p> <p>6. The OCTT responds with a FirmwareStatusNotificationResponse.</p> <p>8. The OCTT responds with a FirmwareStatusNotificationResponse.</p> <p>10. The OCTT responds with a FirmwareStatusNotificationResponse.</p> <p>12. The OCTT responds with a FirmwareStatusNotificationResponse.</p>

Test case name	Secure Firmware Update - Unable to install firmware with ongoing transaction - AllowNewSessionsPendingFirmwareUpdate is false	
	<p>14. The Charging Station sends a FirmwareStatusNotificationRequest.</p> <p><u>Note(s):</u></p> <ul style="list-style-type: none"> - This step is optional if the Charging Station needs to reboot during a firmware update AND the bootloader is unable to send OCPP messages. - If the Charging Station does NOT need to reboot during a firmware update then this step needs to be executed and then proceed to step 22. 	<p>15. The OCTT responds with a FirmwareStatusNotificationResponse.</p>
	<p>16. The Charging Station sends a FirmwareStatusNotificationRequest.</p> <p><u>Note(s):</u></p> <ul style="list-style-type: none"> - This step is optional if the Charging Station needs to reboot during a firmware update AND the bootloader is unable to send OCPP messages. 	<p>17. The OCTT responds with a FirmwareStatusNotificationResponse.</p>
	<p>18. The Charging Station sends a BootNotificationRequest</p>	<p>19. The OCTT responds with a BootNotificationResponse with status Accepted</p>
	<p>20. The Charging Station sends a SecurityEventNotificationRequest with type FirmwareUpdated</p>	<p>21. The OCTT responds with a SecurityEventNotificationResponse</p>
	<p>22. The Charging Station notifies the CSMS about the current state of all connectors.</p>	<p>23. The OCTT responds accordingly.</p>
	<p>24. The Charging Station sends a FirmwareStatusNotificationRequest.</p>	<p>25. The OCTT responds with a FirmwareStatusNotificationResponse.</p>

Test case name	Secure Firmware Update - Unable to install firmware with ongoing transaction - AllowNewSessionsPendingFirmwareUpdate is false
Tool validations	<ul style="list-style-type: none"> * Step 2: Message UpdateFirmwareResponse - status Accepted * Step 3: Message: StatusNotificationRequest - connectorStatus Unavailable Message: NotifyEventRequest - eventData[0].trigger Delta - eventData[0].actualValue "Unavailable" - eventData[0].component.name "Connector" - eventData[0].variable.name "AvailabilityState" * Step 5: Message FirmwareStatusNotificationRequest - status Downloading * Step 7: Message FirmwareStatusNotificationRequest - status Downloaded * Step 9: Message FirmwareStatusNotificationRequest - status SignatureVerified * Step 11: Message FirmwareStatusNotificationRequest - status InstallScheduled * Step 14: Message FirmwareStatusNotificationRequest - status Installing * Step 16: Message FirmwareStatusNotificationRequest - status InstallRebooting
	<ul style="list-style-type: none"> * Step 18: Message BootNotificationRequest - reason FirmwareUpdate * Step 20: Message SecurityEventNotificationRequest - type FirmwareUpdated * Step 22: Message: StatusNotificationRequest - connectorStatus Available Message: NotifyEventRequest - eventData[0].trigger Delta - eventData[0].actualValue "Available" - eventData[0].component.name "Connector" - eventData[0].variable.name "AvailabilityState" * Step 24: Message FirmwareStatusNotificationRequest - status Installed <p>Post scenario validations: N/a</p>

Table 203. Test Case Id: TC_L_16_CS

Test case name	Secure Firmware Update - Able to update firmware with ongoing transaction															
Test case Id	TC_L_16_CS															
Use case Id(s)	L01															
Requirement(s)	L01.FR.01,L01.FR.06,L01.FR.10,L01.FR.20															
System under test	Charging Station															
Description	The CSMS is able to request the Charging Station to securely download and install a new firmware by sending an UpdateFirmwareRequest with a signingCertificate .															
Purpose	To verify if the Charging Station is able to securely download and install a new firmware, while a transaction is ongoing.															
Prerequisite(s)	<ul style="list-style-type: none"> - A file server has been setup according to the (by the Charging Station) supported file transfer protocol(s), indicated by the configuration variable FileTransferProtocols. - The Charging Station is able to update its firmware while a transaction is ongoing. 															
Before (Preparations)	<p>Configuration State: N/a</p> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>EnergyTransferStarted</i></p>															
Main (Test scenario)	<table border="1"> <thead> <tr> <th>Charging Station</th> <th>CSMS</th> </tr> </thead> <tbody> <tr> <td>2. The Charging Station responds with a UpdateFirmwareResponse</td> <td> 1. The OCTT sends a UpdateFirmwareRequest with firmware.installDateTime <Current DateTime - 2 hours> firmware.location <Configured firmware_location> firmware.retrieveDateTime <Current DateTime - 2 hours> firmware.signingCertificate <Configured signingCertificate> firmware.signature <Configured signature> </td> </tr> <tr> <td>3. The Charging Station sends a FirmwareStatusNotificationRequest.</td> <td> 4. The OCTT responds with a FirmwareStatusNotificationResponse. </td> </tr> <tr> <td>5. The Charging Station sends a FirmwareStatusNotificationRequest.</td> <td> 6. The OCTT responds with a FirmwareStatusNotificationResponse. </td> </tr> <tr> <td>7. The Charging Station sends a FirmwareStatusNotificationRequest.</td> <td> 8. The OCTT responds with a FirmwareStatusNotificationResponse. </td> </tr> <tr> <td>9. The Charging Station sends a FirmwareStatusNotificationRequest.</td> <td> 10. The OCTT responds with a FirmwareStatusNotificationResponse. </td> </tr> <tr> <td>11. The Charging Station sends a FirmwareStatusNotificationRequest.</td> <td> 12. The OCTT responds with a FirmwareStatusNotificationResponse. </td> </tr> </tbody> </table>		Charging Station	CSMS	2. The Charging Station responds with a UpdateFirmwareResponse	1. The OCTT sends a UpdateFirmwareRequest with firmware.installDateTime <Current DateTime - 2 hours> firmware.location <Configured firmware_location> firmware.retrieveDateTime <Current DateTime - 2 hours> firmware.signingCertificate <Configured signingCertificate> firmware.signature <Configured signature>	3. The Charging Station sends a FirmwareStatusNotificationRequest .	4. The OCTT responds with a FirmwareStatusNotificationResponse .	5. The Charging Station sends a FirmwareStatusNotificationRequest .	6. The OCTT responds with a FirmwareStatusNotificationResponse .	7. The Charging Station sends a FirmwareStatusNotificationRequest .	8. The OCTT responds with a FirmwareStatusNotificationResponse .	9. The Charging Station sends a FirmwareStatusNotificationRequest .	10. The OCTT responds with a FirmwareStatusNotificationResponse .	11. The Charging Station sends a FirmwareStatusNotificationRequest .	12. The OCTT responds with a FirmwareStatusNotificationResponse .
Charging Station	CSMS															
2. The Charging Station responds with a UpdateFirmwareResponse	1. The OCTT sends a UpdateFirmwareRequest with firmware.installDateTime <Current DateTime - 2 hours> firmware.location <Configured firmware_location> firmware.retrieveDateTime <Current DateTime - 2 hours> firmware.signingCertificate <Configured signingCertificate> firmware.signature <Configured signature>															
3. The Charging Station sends a FirmwareStatusNotificationRequest .	4. The OCTT responds with a FirmwareStatusNotificationResponse .															
5. The Charging Station sends a FirmwareStatusNotificationRequest .	6. The OCTT responds with a FirmwareStatusNotificationResponse .															
7. The Charging Station sends a FirmwareStatusNotificationRequest .	8. The OCTT responds with a FirmwareStatusNotificationResponse .															
9. The Charging Station sends a FirmwareStatusNotificationRequest .	10. The OCTT responds with a FirmwareStatusNotificationResponse .															
11. The Charging Station sends a FirmwareStatusNotificationRequest .	12. The OCTT responds with a FirmwareStatusNotificationResponse .															

Test case name	Secure Firmware Update - Able to update firmware with ongoing transaction
Tool validations	<ul style="list-style-type: none"> * Step 2: Message UpdateFirmwareResponse - status Accepted * Step 3: Message FirmwareStatusNotificationRequest - status Downloading * Step 5: Message FirmwareStatusNotificationRequest - status Downloaded * Step 7: Message FirmwareStatusNotificationRequest - status SignatureVerified * Step 9: Message FirmwareStatusNotificationRequest - status Installing * Step 11: Message FirmwareStatusNotificationRequest - status Installed <p>Post scenario validations: N/a</p>

Table 204. Test Case Id: TC_L_18_CS

Test case name	Secure Firmware Update - Missing firmware signing certificate and signature	
Test case Id	TC_L_18_CS	
Use case Id(s)	L01	
Requirement(s)	N/a	
System under test	Charging Station	
Description	The CSMS is able to request the Charging Station to securely download and install a new firmware by sending an UpdateFirmwareRequest with a signingCertificate .	
Purpose	To verify if the Charging Station is not accepting a non-secure firmware update request, when supporting secure firmware update.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with a UpdateFirmwareResponse	<p>1. The OCTT sends a UpdateFirmwareRequest with firmware.installDateTime <Current DateTime - 2 hours> firmware.location <Configured firmware_location> firmware.retrieveDateTime <Current DateTime - 2 hours> firmware.signingCertificate is omitted firmware.signature is omitted</p>
Tool validations	<ul style="list-style-type: none"> * Step 2: Message UpdateFirmwareResponse - status Rejected OR InvalidCertificate 	
	Post scenario validations: N/a	

2.14. M ISO IEC 15118 CertificateManagement

Table 205. Test Case Id: TC_M_01_CS

Test case name	Install CA certificate - CSMSRootCertificate	
Test case Id	TC_M_01_CS	
Use case Id(s)	M05	
Requirement(s)	M05.FR.01,M05.FR.02	
System under test	Charging Station	
Description	The CSMS is able to request the Charging Station to install new Root CA certificates using the InstallCertificateRequest message.	
Purpose	To verify if the Charging Station is able to install a new CSMSRootCertificate.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	1. Execute Reusable State CertificateInstalled for certificateType CSMSRootCertificate	
	<p>Note(s):</p> <ul style="list-style-type: none"> - When the Charging Station has the following configuration; AdditionalRootCertificateCheck implemented with value true, then a custom CSMSRootCertificate should be used. - When the Charging Station has the following configuration; AdditionalRootCertificateCheck implemented with value false, then the the built-in action to delete the newly installed certificate should be executed. <p>2. Execute Reusable State GetInstalledCertificates for certificateType CSMSRootCertificate</p>	
Tool validations	N/a	
	Post scenario validations: N/a	

Table 206. Test Case Id: TC_M_02_CS

Test case name	Install CA certificate - ManufacturerRootCertificate	
Test case Id	TC_M_02_CS	
Use case Id(s)	M05	
Requirement(s)	M05.FR.01,M05.FR.02	
System under test	Charging Station	
Description	The CSMS is able to request the Charging Station to install new Root CA certificates using the InstallCertificateRequest message.	
Purpose	To verify if the Charging Station is able to install a new ManufacturerRootCertificate.	
Prerequisite(s)	The Charging Station does NOT have the following configuration; AdditionalRootCertificateCheck is implemented with value true	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): N/a	
Main (Test scenario)	Charging Station <ol style="list-style-type: none"> 1. Execute Reusable State CertificateInstalled for certificateType <i>ManufacturerRootCertificate</i> 2. Execute Reusable State GetInstalledCertificates for certificateType <i>ManufacturerRootCertificate</i> 	CSMS
Tool validations	N/a Post scenario validations: N/a	

Table 207. Test Case Id: TC_M_07_CS

Test case name	Install CA certificate - Rejected - Certificate invalid	
Test case Id	TC_M_07_CS	
Use case Id(s)	M05	
Requirement(s)	M05.FR.01,M05.FR.07	
System under test	Charging Station	
Description	The CSMS is able to request the Charging Station to install new Root CA certificates using the InstallCertificateRequest message.	
Purpose	To verify if the Charging Station is able to reject an invalid certificate.	
Prerequisite(s)	The Charging Station does NOT have the following configuration; AdditionalRootCertificateCheck is implemented with value true	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with a InstallCertificateResponse	1. The OCTT sends a InstallCertificateRequest with certificateType is CSMSRootCertificate certificate is <Generated Expired Certificate>
	4. The Charging Station responds with a GetInstalledCertificateIdsResponse	3. The OCTT sends a GetInstalledCertificateIdsRequest with certificateType is CSMSRootCertificate
Tool validations	<ul style="list-style-type: none"> * Step 2: Message: InstallCertificateResponse - status must be <i>Rejected</i> * Step 4: Message: GetInstalledCertificateIdsResponse - status must be <i>Accepted</i> - certificateHashDataChain must NOT contain an entry with following values: Note: Order does not matter. - certificateType is CSMSRootCertificate - certificateHashData contains <HashData from configured new CSMS Root certificate> <p>Post scenario validations: N/a</p>	

Table 208. Test Case Id: TC_M_09_CS

Test case name	Install CA certificate - AdditionalRootCertificateCheck - Rejected	
Test case Id	TC_M_09_CS	
Use case Id(s)	M05	
Requirement(s)	M05.FR.10,M05.FR.11	
System under test	Charging Station	
Description	The CSMS is able to request the Charging Station to install new Root CA certificates using the InstallCertificateRequest message.	
Purpose	To verify if the Charging Station is able to reject installing a new CSMSRootCertificate that is not signed by the old CSMSRootCertificate, while additional security measures for installing a root certificate is active.	
Prerequisite(s)	- The Charging Station has the configuration variable AdditionalRootCertificateCheck implemented with value <i>true</i>	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with a InstallCertificateResponse	1. The OCTT sends a InstallCertificateRequest with certificateType is <i>CSMSRootCertificate</i> certificate is <Configured CSMSRootCertificate> <u>Note(s):</u> - CSMSRootCertificate must have not been signed by old certificate.
	4. The Charging Station responds with a GetInstalledCertificateIdsResponse	3. The OCTT sends a GetInstalledCertificateIdsRequest with certificateType is <i>CSMSRootCertificate</i>
Tool validations	<ul style="list-style-type: none"> * Step 2: Message: InstallCertificateResponse - status must be <i>Rejected</i> * Step 4: Message: GetInstalledCertificateIdsResponse - status must be <i>Accepted</i> - certificateHashDataChain must contain one entry with following values: - certificateType is <i>CSMSRootCertificate</i> - certificateHashData contains <HashData from configured old CSMS Root certificate> <p>Post scenario validations: N/a</p>	

Table 209. Test Case Id: TC_M_30_CS

Test case name	Install CA certificate - AdditionalRootCertificateCheck - Reconnect using new CSMS Root - Success	
Test case Id	TC_M_30_CS	
Use case Id(s)	M05	
Requirement(s)	M05.FR.13	
System under test	Charging Station	
Description	The CSMS is able to request the Charging Station to install new Root CA certificates using the InstallCertificateRequest message.	
Purpose	To verify if the Charging Station is able to reconnect to the CSMS, while using a new CSMS Root certificate.	
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station has the configuration variable AdditionalRootCertificateCheck implemented with value true - The at the OCTT configured new CSMSRootCertificate must be signed by the old CSMS Root certificate. 	
Before (Preparations)	<p>Configuration State: N/a</p> <p>Memory State: <i>CertificateInstalled</i> for certificateType CSMSRootCertificate and certificate <<i>Configured new CSMS Root certificate 2</i>></p> <p>If security profile 3 is enabled, then: <i>RenewChargingStationCertificate</i> for certificateType ChargingStationCertificate</p> <p>Reusable State(s): N/a</p>	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with a ResetResponse	1. The OCTT sends a ResetRequest with type OnIdle
	4. During the TLS handshake the Charging Station validates the CSMS certificate.	3. During the TLS handshake the OCTT provides a CSMS certificate which is signed by the < <i>Configured new CSMS Root certificate</i> >
	Note(s): - This connection attempt must succeed.	
	5. Execute Reusable State Booted	
	7. The Charging Station responds with a GetInstalledCertificateIdsResponse	6. The OCTT sends a GetInstalledCertificateIdsRequest with certificateType is CSMSRootCertificate
Tool validations	<ul style="list-style-type: none"> * Step 2: Message ResetResponse - status Accepted * Step 7: Message: GetInstalledCertificateIdsResponse - status must be Accepted - certificateHashDataChain must NOT contain an entry with following values: - certificateType is CSMSRootCertificate - certificateHashData contains <<i>HashData from configured old CSMS Root certificate</i>> Post scenario validations: - N/a 	

Table 210. Test Case Id: TC_M_31_CS

Test case name	Install CA certificate - AdditionalRootCertificateCheck - Reconnect using new CSMS Root - Fallback mechanism	
Test case Id	TC_M_31_CS	
Use case Id(s)	M05	
Requirement(s)	M05.FR.14	
System under test	Charging Station	
Description	The CSMS is able to request the Charging Station to install new Root CA certificates using the InstallCertificateRequest message.	
Purpose	To verify if the Charging Station is able to reconnect to the CSMS using the old CSMS Root certificate, when validating the CSMS certificate using the new CSMS Root certificate fails.	
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station has the configuration variable AdditionalRootCertificateCheck implemented with value true - The at the OCTT configured new CSMSRootCertificate must be signed by the old CSMS Root certificate. 	
Before (Preparations)	<p>Configuration State: N/a</p> <p>Memory State: <i>CertificateInstalled</i> for certificateType CSMSRootCertificate and certificate <Configured (new) CSMS Root certificate 2></p> <p>Reusable State(s): N/a</p>	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with a ResetResponse	1. The OCTT sends a ResetRequest with type OnIdle
	4. During the TLS handshake the Charging Station validates the CSMS certificate.	3. During the TLS handshake the OCTT provides a CSMS certificate which is signed by the <Configured old CSMS Root certificate>
	Note(s): - This connection attempt fails, because the Charging Station will use the new CSMS Root certificate to validate the CSMS certificate.	
	5. The Charging Station re-validates the CSMS certificate.	
	Note(s): - This connection attempt succeeds, because the Charging Station will now use the old CSMS Root certificate to validate the CSMS certificate.	
	6. Execute Reusable State Booted	
	8. The Charging Station responds with a GetInstalledCertificateIdsResponse	7. The OCTT sends a GetInstalledCertificateIdsRequest with certificateType is CSMSRootCertificate

Test case name	Install CA certificate - AdditionalRootCertificateCheck - Reconnect using new CSMS Root - Fallback mechanism
Tool validations	<ul style="list-style-type: none"> * Step 2: Message ResetResponse - status Accepted * Step 8: Message: GetInstalledCertificateIdsResponse - status must be <i>Accepted</i> - certificateHashDataChain must contain an entry with following values: - certificateType is <i>CSMSRootCertificate</i> - certificateHashData contains <i><HashData from configured old CSMS Root certificate></i>
	Post scenario validations: <ul style="list-style-type: none"> - N/a

Table 211. Test Case Id: TC_M_12_CS

Test case name	Retrieve certificates from Charging Station - CSMSRootCertificate	
Test case Id	TC_M_12_CS	
Use case Id(s)	M03	
Requirement(s)	M03.FR.01,M03.FR.03,M03.FR.04	
System under test	Charging Station	
Description	The CSMS is able to retrieve the certificates installed at the Charging Station using the GetInstalledCertificatesRequest message.	
Purpose	To verify if the Charging Station is able to provide the hashData from all stored CSMSRootCertificates.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: <i>CertificateInstalled</i> from certificateType CSMSRootCertificate	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	1. Execute Reusable State <i>GetInstalledCertificates</i> for certificateType CSMSRootCertificate	
Tool validations	N/a	
	Post scenario validations: N/a	

Table 212. Test Case Id: TC_M_13_CS

Test case name	Retrieve certificates from Charging Station - ManufacturerRootCertificate	
Test case Id	TC_M_13_CS	
Use case Id(s)	M03	
Requirement(s)	M03.FR.01,M03.FR.03,M03.FR.04	
System under test	Charging Station	
Description	The CSMS is able to retrieve the certificates installed at the Charging Station using the GetInstalledCertificatesRequest message.	
Purpose	To verify if the Charging Station is able to provide the hashData from all stored ManufacturerRootCertificate.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: <i>CertificateInstalled</i> from certificateType <i>ManufacturerRootCertificate</i>	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	1. Execute Reusable State <i>GetInstalledCertificates</i> for certificateType <i>ManufacturerRootCertificate</i>	
Tool validations	N/a	
	Post scenario validations: N/a	

Table 213. Test Case Id: TC_M_17_CS

Test case name	Retrieve certificates from Charging Station - CSMSRootCertificate & ManufacturerRootCertificate	
Test case Id	TC_M_17_CS	
Use case Id(s)	M03	
Requirement(s)	M03.FR.01,M03.FR.03,M03.FR.04	
System under test	Charging Station	
Description	The CSMS is able to retrieve the certificates installed at the Charging Station using the GetInstalledCertificatesRequest message.	
Purpose	To verify if the Charging Station is able to provide the hashData from all stored CSMSRootCertificates and ManufacturerRootCertificates	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: <i>CertificateInstalled</i> from certificateType CSMSRootCertificate <i>CertificateInstalled</i> from certificateType ManufacturerRootCertificate	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	1. Execute Reusable State <i>GetInstalledCertificates</i> for certificateType CSMSRootCertificate AND ManufacturerRootCertificate	
Tool validations	N/a	
	Post scenario validations: N/a	

Table 214. Test Case Id: TC_M_18_CS

Test case name	Retrieve certificates from Charging Station - All certificateTypes	
Test case Id	TC_M_18_CS	
Use case Id(s)	M03	
Requirement(s)	M03.FR.01,M03.FR.03,M03.FR.04	
System under test	Charging Station	
Description	The CSMS is able to retrieve the certificates installed at the Charging Station using the GetInstalledCertificateIdsRequest message.	
Purpose	To verify if the Charging Station is able to provide the hashData from all stored certificates	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: <i>CertificateInstalled</i> from certificateType CSMSRootCertificate <i>CertificateInstalled</i> from certificateType ManufacturerRootCertificate	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with a GetInstalledCertificateIdsResponse	1. The OCTT sends a GetInstalledCertificateIdsRequest With certificateType is omitted.
Tool validations	<ul style="list-style-type: none"> * Step 2: <p>Message: GetInstalledCertificateIdsResponse</p> <ul style="list-style-type: none"> - status must be Accepted - certificateHashDataChain must contain the following two entries with following values: Note: Order does not matter. <p>Entry 1:</p> <ul style="list-style-type: none"> - certificateHashDataChain[0].certificateType is CSMSRootCertificate - certificateHashDataChain[0].certificateHashData contains <HashData from configured new CSMS Root certificate> <p>Entry 2:</p> <ul style="list-style-type: none"> - certificateHashDataChain[1].certificateType is ManufacturerRootCertificate - certificateHashDataChain[1].certificateHashData contains <HashData from configured new Manufacturer Root certificate> 	
	Post scenario validations: N/a	

Table 215. Test Case Id: TC_M_19_CS

Test case name	Retrieve certificates from Charging Station - No matching certificate found	
Test case Id	TC_M_19_CS	
Use case Id(s)	M03	
Requirement(s)	M03.FR.01,M03.FR.02	
System under test	Charging Station	
Description	The CSMS is able to retrieve the certificates installed at the Charging Station using the GetInstalledCertificateIdsRequest message.	
Purpose	To verify if the Charging Station is able to respond that it did not find any certificate of the requested certificateType.	
Prerequisite(s)	The Charging Station does not have a M0RootCertificate installed.	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): N/a	
Main (Test scenario)	Charging Station 2. The Charging Station responds with a GetInstalledCertificateIdsResponse	CSMS 1. The OCTT sends a GetInstalledCertificateIdsRequest With certificateType is <i>M0RootCertificate</i>
Tool validations	* Step 2: Message: GetInstalledCertificateIdsResponse - status must be <i>NotFound</i> - certificateHashDataChain must be omitted. Post scenario validations: N/a	

Table 216. Test Case Id: TC_M_20_CS

Test case name	Delete a certificate from a Charging Station - Success	
Test case Id	TC_M_20_CS	
Use case Id(s)	M04	
Requirement(s)	M04.FR.01,M04.FR.02	
System under test	Charging Station	
Description	The CSMS is able to request the Charging Station to delete an installed certificate using the DeleteCertificateRequest message.	
Purpose	To verify if the Charging Station is able to delete an installed certificate.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): GetInstalledCertificates with certificateType CSMSRootCertificate CertificateInstalled with certificateType CSMSRootCertificate (When no certificate is returned at GetInstalledCertificates)	
Main (Test scenario)	Charging Station	CSMS
	1. Execute Reusable State GetInstalledCertificates with certificateType CSMSRootCertificate	
	3. The Charging Station responds with a DeleteCertificateResponse	2. The OCTT sends a DeleteCertificateRequest with certificateHashData contains <Returned certificateHashData at step 1>
	4. Execute Reusable State GetInstalledCertificates with certificateType CSMSRootCertificate	
Tool validations	<ul style="list-style-type: none"> * Step 1: - Certificate that is going to be deleted is present. * Step 3: Message: DeleteCertificateResponse - status must be Accepted * Step 4: - Certificate that should be deleted is not present anymore. 	
	Post scenario validations: N/a	

Table 217. Test Case Id: TC_M_22_CS

Test case name	Delete a certificate from a Charging Station - No matching certificate found	
Test case Id	TC_M_22_CS	
Use case Id(s)	M04	
Requirement(s)	M04.FR.01,M04.FR.04	
System under test	Charging Station	
Description	The CSMS is able to request the Charging Station to delete an installed certificate using the DeleteCertificateRequest message.	
Purpose	To verify if the Charging Station is able to respond that no certificate is installed that matches the provided certificateHashData.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	1. Execute Reusable State GetInstalledCertificates with certificateType CSMSRootCertificate.	
	3. The Charging Station responds with a DeleteCertificateResponse	2. The OCTT sends a DeleteCertificateRequest with certificateHashData is <certificateHashData from unknown certificate>
Tool validations	<ul style="list-style-type: none"> * Step 3: Message: DeleteCertificateResponse - status must be <i>NotFound</i> 	
	Post scenario validations: N/a	

Table 218. Test Case Id: TC_M_23_CS

Test case name	Delete a certificate from a Charging Station - Unable to delete the Charging Station Certificate	
Test case Id	TC_M_23_CS	
Use case Id(s)	M04	
Requirement(s)	M04.FR.01,M04.FR.06	
System under test	Charging Station	
Description	The CSMS is able to request the Charging Station to delete an installed certificate using the DeleteCertificateRequest message.	
Purpose	To verify if the Charging Station does NOT allow the deletion of the Charging Station certificate.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): <i>RenewChargingStationCertificate</i> for certificateType <i>ChargingStationCertificate</i>	
Main (Test scenario)	Charging Station	CSMS
	1. Execute Reusable State <i>GetInstalledCertificates</i> with certificateType omitted.	
	3. The Charging Station responds with a DeleteCertificateResponse	2. The OCTT sends a DeleteCertificateRequest with certificateHashData is <certificateHashData from the generated ChargingStationCertificate at before.>
Tool validations	<ul style="list-style-type: none"> * Step 3: Message: DeleteCertificateResponse - status must be <i>NotFound</i> OR <i>Failed</i> 	
	Post scenario validations: N/a	

2.15. N Diagnostics

Table 219. Test Case Id: TC_N_25_CS

Test case name	Retrieve Log Information - Diagnostics Log - Success	
Test case Id	TC_N_25_CS	
Use case Id(s)	N01	
Requirement(s)	N01.FR.01, N01.FR.02, N01.FR.04, N01.FR.07, N01.FR.08, N01.FR.09, N01.FR.13	
System under test	Charging Station	
Description	This test case covers the functionality of getting log information from a Charging Station. The CSMS can request a Charging Station to upload a file with log information to a given location (URL). The format of this log file is not prescribed. The Charging Station successfully uploads a log file and gives information about the status of the upload by sending status notifications to the CSMS.	
Purpose	To verify if the Charging station is able to successfully upload a log as described at the OCPP specification.	
Prerequisite(s)	<ul style="list-style-type: none"> - Charging Station has log information available. - A diagnostics logging server has been setup supporting one of the file transfer protocols supported by the Charging Station (This is configured at the configuration variable FileTransferProtocols). 	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with a GetLogResponse	1. The OCTT sends a GetLogRequest with logType DiagnosticsLog
	Note(s): - <i>Charging Station is uploading log file</i>	
	3. The Charging Station sends a LogStatusNotificationRequest	4. The OCTT responds with a LogStatusNotificationResponse
	Note(s): - <i>Log file is uploaded</i>	
	5. The Charging Station sends a LogStatusNotificationRequest	6. The OCTT responds with a LogStatusNotificationResponse
Tool validations	<ul style="list-style-type: none"> * Step 2: Message GetLogResponse - status Accepted - filename not omitted AND not empty * Step 3: Message LogStatusNotificationRequest - status Uploading - requestId Same Id as the GetLogRequest * Step 5: Message LogStatusNotificationRequest - status Uploaded - requestId Same Id as the GetLogRequest <p>Post scenario validations: - N/a</p>	

Table 220. Test Case Id: TC_N_26_CS

Test case name	Retrieve Log Information - Diagnostics Log - Upload failed	
Test case Id	TC_N_26_CS	
Use case Id(s)	N01	
Requirement(s)	N01.FR.01, N01.FR.02, N01.FR.04, N01.FR.07, N01.FR.08, N01.FR.10, N01.FR.13	
System under test	Charging Station	
Description	This test case covers the functionality of getting log information from a Charging Station. The CSMS can request a Charging Station to upload a file with log information to a given location (URL). The format of this log file is not prescribed. The Charging Station unsuccessfully uploads a log file and gives information about the status of the upload by sending status notifications to the CSMS.	
Purpose	To verify if the Charging Station is able to correctly communicate with the CSMS after failing to upload a log as described at the OCPP specification.	
Prerequisite(s)	- Charging Station has log information available.	
Before (Preparations)	Configuration State: N/a	
	Memory State: Charging Station has log information available.	
	Charging State: N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with a GetLogResponse	1. The OCTT sends a GetLogRequest with - logType <i>DiagnosticsLog</i> - retries 3 - retryInterval <Configured retryInterval>
	3. The Charging Station sends a LogStatusNotificationRequest	4. The OCTT responds with a LogStatusNotificationResponse
	5. The Charging Station sends a LogStatusNotificationRequest	6. The OCTT responds with a LogStatusNotificationResponse
	Note(s): - Steps 3 & 4 are optional after the first attempt. - The Charging Station will perform step (3,) 5, three times with <Configured retryInterval> seconds in between.	

Test case name	Retrieve Log Information - Diagnostics Log - Upload failed
Tool validations	<ul style="list-style-type: none"> * Step 1: Message GetLogResponse - status Accepted * Step 3: Message LogStatusNotificationRequest - status Uploading - requestId Same Id as the GetLogRequest * Step 5: Message LogStatusNotificationRequest - status UploadFailure - requestId Same Id as the GetLogRequest OR Message LogStatusNotificationRequest - status BadMessage - requestId Same Id as the GetLogRequest OR Message LogStatusNotificationRequest - status PermissionDenied - requestId Same Id as the GetLogRequest OR Message LogStatusNotificationRequest - status NotSupportedOperation - requestId Same Id as the GetLogRequest * The time between the first LogStatusNotificationRequest Uploading and the last LogStatusNotificationRequest UploadFailure/BadMessage/PermissionDenied/NotSupportedOperation equals $(3 * \text{<Configured retryInterval>})$
Post scenario validations:	<ul style="list-style-type: none"> - N/a

Table 221. Test Case Id: TC_N_27_CS

Test case name	Get Customer Information - Accepted + data	
Test case Id	TC_N_27_CS	
Use case Id(s)	N09	
Requirement(s)	N09.FR.02, N09.FR.05	
System under test	Charging Station	
Description	The CSMS sends a message to the Charging Station to retrieve IdToken customer information, for example to be compliant with local privacy laws. The Charging Station notifies the CSMS by sending one or more reports.	
Purpose	To verify if the Charging Station accepts the request and correctly sends the information as described at the OCPP specification.	
Prerequisite(s)	The Charging Station needs to support Local Authorization and either the Local Authorization List or Authorization Cache.	
Before (Preparations)	<p>Configuration State: LocalAuthListCtrlr.Enabled is set to <i>true</i> AuthCtrlr.LocalPreAuthorize is set to <i>true</i> AuthCacheCtrlr.Enabled is set to <i>true</i></p> <p>Memory State: <i>IdTokenCached</i> for <Configured valid IdToken fields> (If implemented) <i>IdTokenLocalAuthList</i> for <Configured valid IdToken fields> (If implemented)</p> <p>Charging State: State is <i>Authorized</i> (local) State is <i>ParkingBayUnoccupied</i></p>	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with a CustomerInformationResponse	1. The OCTT sends a CustomerInformationRequest with - report true - idToken <Configured valid idToken fields>
	3. The Charging Station sends a NotifyCustomerInformationRequest	4. The OCTT responds with a NotifyCustomerInformationResponse .
	<u>Note(s):</u> - If tbc is True at Step 3 then step 3 and 4 will be repeated	
Tool validations	<ul style="list-style-type: none"> * Step 2: Message CustomerInformationResponse - status Accepted * Step 3: Message NotifyCustomerInformationRequest - data Not empty <p>Post scenario validations: - All report parts have been received</p>	

Table 222. Test Case Id: TC_N_28_CS

Test case name	Get Customer Information - Accepted + no data	
Test case Id	TC_N_28_CS	
Use case Id(s)	N09	
Requirement(s)	N09.FR.02, N09.FR.06	
System under test	Charging Station	
Description	The CSMS sends a message to the Charging Station to retrieve IdToken customer information, for example to be compliant with local privacy laws. The Charging Station notifies the CSMS by sending one or more reports.	
Purpose	To verify if the Charging Station accepts the request and correctly respond when it couldn't find the right information as described at the OCPP specification.	
Prerequisite(s)	Charging Station has no customer information available of <Configured valid idToken fields>	
Before (Preparations)	Configuration State: N/a	
	Memory State: The CSMS requests the CS to clear the customerInformation for idToken <Configured valid idToken fields>	
	Charging State: N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with a CustomerInformationResponse	1. The OCTT sends a CustomerInformationRequest with - report true - idToken <Configured valid idToken fields>
	3. The Charging Station sends a NotifyCustomerInformationRequest	4. The OCTT responds with a NotifyCustomerInformationResponse .
Tool validations	<ul style="list-style-type: none"> * Step 2: Message CustomerInformationResponse - status Accepted * Step 3: Message NotifyCustomerInformationRequest - tbc Not true <p>Post scenario validations: - A message is sent indicating that no data is found</p>	

Table 223. Test Case Id: TC_N_30_CS

Test case name	Clear Customer Information - Clear and report + data	
Test case Id	TC_N_30_CS	
Use case Id(s)	N10	
Requirement(s)	N10.FR.01, N10.FR.03	
System under test	Charging Station	
Description	The CSMS sends a message to the Charging Station to clear (and retrieve) IdToken customer information, for example to be compliant with local privacy laws. The Charging Station notifies the CSMS by sending one or more reports.	
Purpose	To verify if the Charging Station accepts the request and removes all customer related data (except from LocalList) and sent notifies as described at the OCPP specification.	
Prerequisite(s)	The Charging Station needs to support Local Authorization and either the Local Authorization List or Authorization Cache.	
Before (Preparations)	<p>Configuration State: LocalAuthListCtrlr.Enabled is set to <i>true</i> AuthCtrlr.LocalPreAuthorize is set to <i>true</i> AuthCacheCtrlr.Enabled is set to <i>true</i></p> <p>Memory State: <i>IdTokenCached</i> for <Configured valid IdToken fields> (If implemented) <i>IdTokenLocalAuthList</i> for <Configured valid IdToken fields> (If implemented)</p> <p>Charging State: State is <i>Authorized</i> (local) State is <i>ParkingBayUnoccupied</i></p>	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with a CustomerInformationResponse	1. The OCTT sends a CustomerInformationRequest with - report true AND - clear true AND - idToken <Configured valid idToken fields>
	3. The Charging Station sends a NotifyCustomerInformationRequest	4. The OCTT responds with a NotifyCustomerInformationResponse .
	<u>Note(s):</u> - If tbc is True at Step 3 then step 3 and 4 will be repeated	6. The Charging Station responds with a CustomerInformationResponse
	5. The OCTT sends a CustomerInformationRequest with - report true AND - idToken <Configured valid idToken fields>	7. The Charging Station sends a NotifyCustomerInformationRequest
		8. The OCTT responds with a NotifyCustomerInformationResponse .
<u>Note(s):</u> - Step is optional and only expected when status is Accepted at Step 6		Tool validations
* Step 2: Message CustomerInformationResponse - status Accepted * Step 3: Message NotifyCustomerInformationRequest - data Not empty * Step 8: Message NotifyCustomerInformationRequest - tbc Not true		

Table 224. Test Case Id: TC_N_31_CS

Test case name	Clear Customer Information - Clear and report + no data	
Test case Id	TC_N_31_CS	
Use case Id(s)	N10	
Requirement(s)	N10.FR.01, N10.FR.04	
System under test	Charging Station	
Description	The CSMS sends a message to the Charging Station to clear (and retrieve) idToken customer information, for example to be compliant with local privacy laws. The Charging Station notifies the CSMS by sending one or more reports.	
Purpose	To verify if the Charging Station accepts the request and correctly respond when it couldn't find the right information as described at the OCPP specification.	
Prerequisite(s)	Charging Station has no customer information available of <Configured valid idToken fields>	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with a CustomerInformationResponse	1. The OCTT sends a CustomerInformationRequest with <ul style="list-style-type: none">- report true AND- clear true AND- idToken <Configured valid idToken fields>
	3. The Charging Station sends a NotifyCustomerInformationRequest	4. The OCTT responds with a NotifyCustomerInformationResponse .
Tool validations	* Step 2: Message CustomerInformationResponse - status Accepted	
	Post scenario validations: - A message is send indicating that no data is found	

Table 225. Test Case Id: TC_N_32_CS

Test case name	Clear Customer Information - Clear and no report	
Test case Id	TC_N_32_CS	
Use case Id(s)	N10	
Requirement(s)	N10.FR.01, N10.FR.06	
System under test	Charging Station	
Description	The CSMS sends a message to the Charging Station to clear (and retrieve) idToken customer information, for example to be compliant with local privacy laws. The Charging Station notifies the CSMS by sending one or more reports.	
Purpose	To verify if the Charging Station accepts the request and removes all customer related data (except from LocalList) and sent one notify as described at the OCPP specification.	
Prerequisite(s)	Charging Station has no customer information available of <Configured valid idToken fields>	
Before (Preparations)	Configuration State: N/a Memory State: N/a Charging State: N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with a CustomerInformationResponse	1. The OCTT sends a CustomerInformationRequest with - report false AND - clear true AND - idToken <Configured valid idToken fields>
	3. The Charging Station sends a NotifyCustomerInformationRequest	4. The OCTT responds with a NotifyCustomerInformationResponse .
Tool validations	* Step 2: Message CustomerInformationResponse - status Accepted Post scenario validations: - A message is send indicating that the data is cleared	

Table 226. Test Case Id: TC_N_62_CS

Test case name	Clear Customer Information - Clear and report - customerIdentifier	
Test case Id	TC_N_62_CS	
Use case Id(s)	N10	
Requirement(s)	N10.FR.01, N10.FR.03	
System under test	Charging Station	
Description	The CSMS sends a message to the Charging Station to clear (and retrieve) raw customer information, for example to be compliant with local privacy laws. The Charging Station notifies the CSMS by sending one or more reports.	
Purpose	To verify if the Charging Station accepts the request and removes all customer related data (except from LocalList) and sent notifies as described at the OCPP specification.	
Prerequisite(s)	The Charging Station needs to support retrieving / deleting CustomerInformation - CustomerIdentifier.	
Before (Preparations)	Configuration State: N/a	
	Memory State: The tester needs manually store the <Configured CustomerIdentifier> at the Charging Station.	
	Charging State: N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with a CustomerInformationResponse	1. The OCTT sends a CustomerInformationRequest with - report true AND - clear true AND - customerIdentifier <Configured customerIdentifier>
	3. The Charging Station sends a NotifyCustomerInformationRequest	4. The OCTT responds with a NotifyCustomerInformationResponse
	<u>Note(s):</u> - If tbc is True at Step 3 then step 3 and 4 will be repeated	
	6. The Charging Station responds with a CustomerInformationResponse	5. The OCTT sends a CustomerInformationRequest with - report true AND - clear false AND - customerIdentifier <Configured customerIdentifier>
	7. The Charging Station sends a NotifyCustomerInformationRequest	8. The OCTT responds with a NotifyCustomerInformationResponse
	<u>Note(s):</u> - If tbc is True at Step 7 then step 7 and 8 will be repeated	
Tool validations	<ul style="list-style-type: none"> * Step 2: Message CustomerInformationResponse - status Accepted * Step 3: Message NotifyCustomerInformationRequest - data Not empty * Step 6: Message CustomerInformationResponse - status Accepted * Step 7: Message NotifyCustomerInformationRequest - data empty <p>Post scenario validations: - All report parts have been received</p>	

Table 227. Test Case Id: TC_N_35_CS

Test case name	Retrieve Log Information - Security Log - Success	
Test case Id	TC_N_35_CS	
Use case Id(s)	N01	
Requirement(s)	N01.FR.01, N01.FR.02, N01.FR.03, N01.FR.07, N01.FR.08, N01.FR.09, N01.FR.13	
System under test	Charging Station	
Description	This test case covers the functionality of getting log information from a Charging Station. The CSMS can request a Charging Station to upload a file with log information to a given location (URL). The format of this log file is not prescribed. The Charging Station successfully uploads a log file and gives information about the status of the upload by sending status notifications to the CSMS.	
Purpose	To verify if the Charging station is able to successfully upload a log as described at the OCPP specification.	
Prerequisite(s)	Charging Station supports Monitoring	
Before (Preparations)	<p>Configuration State: N/a</p> <p>Memory State: Charging Station has log information available.</p> <p>Charging State: N/a</p>	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with a GetLogResponse	1. The OCTT sends a GetLogRequest with logType SecurityLog
	Note(s): - <i>Charging Station is uploading log file</i>	
	3. The Charging Station sends a LogStatusNotificationRequest	4. The OCTT responds with a LogStatusNotificationResponse .
	Note(s): - <i>Log file is uploaded</i>	
	5. The Charging Station sends a LogStatusNotificationRequest	6. The OCTT responds with a LogStatusNotificationResponse .
Tool validations	<ul style="list-style-type: none"> * Step 2: Message GetLogResponse - status Accepted * Step 3: Message LogStatusNotificationRequest - status Uploading - requestId Same Id as the GetLogRequest * Step 5: Message LogStatusNotificationRequest - status Uploaded - requestId Same Id as the GetLogRequest <p>Post scenario validations: - N/a</p>	

Table 228. Test Case Id: TC_N_36_CS

Test case name	Retrieve Log Information - Second Request	
Test case Id	TC_N_36_CS	
Use case Id(s)	N01	
Requirement(s)	N01.FR.01, N01.FR.02, N01.FR.03, N01.FR.07, N01.FR.08, N01.FR.09, N01.FR.12, N01.FR.13	
System under test	Charging Station	
Description	This test case covers the functionality of getting log information from a Charging Station. The CSMS can request a Charging Station to upload a file with log information to a given location (URL). The format of this log file is not prescribed. The Charging Station successfully uploads a log file and gives information about the status of the upload by sending status notifications to the CSMS.	
Purpose	To verify if the Charging station is able to successfully start/cancel a upload on a second request as described at the OCPP specification.	
Prerequisite(s)	Charging Station supports Monitoring	
Before (Preparations)	<p>Configuration State: N/a</p> <p>Memory State: Charging Station has log information available of <<i>Configured logType</i>>.</p> <p>Charging State: N/a</p>	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with a GetLogResponse	1. The OCTT sends a GetLogRequest with logType < <i>Configured logType</i> >
	Note(s): - <i>Charging Station is uploading log file</i>	
	3. The Charging Station sends a LogStatusNotificationRequest	4. The OCTT responds with a LogStatusNotificationResponse .
	Note(s): - <i>Charging Station cancels uploading the first log file</i>	
	6. The Charging Station responds with a GetLogResponse	5. The OCTT sends a GetLogRequest with logType < <i>Configured logType</i> >
	7. The Charging Station sends a LogStatusNotificationRequest	8. The OCTT responds with a LogStatusNotificationResponse .
	Note(s): - <i>Charging Station is uploading log file</i>	
	9. The Charging Station sends a LogStatusNotificationRequest	10. The OCTT responds with a LogStatusNotificationResponse .
	Note(s): - <i>Log file is uploaded</i>	
	11. The Charging Station sends a LogStatusNotificationRequest	12. The OCTT responds with a LogStatusNotificationResponse .

Test case name	Retrieve Log Information - Second Request
Tool validations	<ul style="list-style-type: none"> * Step 2: Message GetLogResponse - status Accepted * Step 3: Message LogStatusNotificationRequest - status Uploading - requestId Same Id as the GetLogRequest * Step 6: Message GetLogResponse - status AcceptedCanceled * Step 7: Message LogStatusNotificationRequest - status AcceptedCanceled * Step 9: Message LogStatusNotificationRequest - status Uploading - requestId Same Id as the GetLogRequest * Step 11: Message LogStatusNotificationRequest - status Uploaded - requestId Same Id as the GetLogRequest <p>Post scenario validations: - N/a</p>

2.16. O Display Message

This section is intentionally blank, this will be added in a later version.

2.17. P DataTransfer

Table 229. Test Case Id: TC_P_01_CS

Test case name	Data Transfer to the Charging Station - Rejected / Unknown VendorId / Unknown MessageId	
Test case Id	TC_P_01_CS	
Use case Id(s)	P01	
Requirement(s)	P01.FR.05, P01.FR.06	
System under test	Charging Station	
Description	The DataTransfer message to send information for functions that are not supported by OCPP.	
Purpose	To verify whether the Charging Station is able to handle receiving a DataTransferRequest, even if it does not support any vendor-specific implementations.	
Prerequisite(s)	The configured vendorId should not be implemented and the configured messageId should be unused.	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with a DataTransferResponse	1. The OCTT sends a DataTransferRequest with vendorId org.openchargealliance.octt and messageId <Configured messageId>
Tool validations	<ul style="list-style-type: none"> * Step 2: Message: DataTransferResponse <ul style="list-style-type: none"> - status must be <i>UnknownVendorId</i> OR <i>UnknownMessageId</i> OR <i>Rejected</i> (<i>Rejected</i> will also be allowed, because there are implementers that like to just reject the message when the Charging Station does not support any vendor-specific features.) 	
	Post scenario validations: N/a	

Table 230. Test Case Id: TC_P_03_CS

Test case name	CustomData - Receive custom data	
Test case Id	TC_P_03_CS	
Use case Id(s)	N/a	
Requirement(s)	N/a	
System under test	Charging Station	
Description	Checks if the CS is able to receive custom data.	
Purpose	To verify whether the CS is able to handle receiving custom data.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The Charging Station responds with SetVariablesResponse	1. OCTT sends SetVariablesRequest with: - variable.name = "OfflineThreshold" - component.name = "OCPPCommCtrlr" - attributeValue = "200" - attributeType is Actual
	4. The Charging Station responds with GetVariablesResponse	3. OCTT sends GetVariablesRequest with: - variable.name = "OfflineThreshold" - component.name = "OCPPCommCtrlr" - attributeType is Actual
Tool validations	<ul style="list-style-type: none"> * Step 2: Message: SetVariablesResponse - setVariableResult[0].attributeStatus Accepted * Step 4: Message: GetVariablesResponse - getVariableResult[0].attributeStatus Accepted - getVariableResult[0].attributeType Actual or omitted - getVariableResult[0].attributeValue 200 <p>Post scenario validations: - N/a</p>	

2.18. Reusable states

Testcases can refer to a reusable state at the before or main stage. The steps described at the reusable state will be executed and then it will return to the testcase that called the reusable state.

Table 231. Reusable State: Booting

State	Booting	
System under test	Charging Station	
Description	This state will prepare the Charging Station, so that it is still booting. The connection has not been setup yet.	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Scenario)	Charging Station	CSMS
	2. The Charging Station responds with a ResetResponse	1. The OCTT sends a ResetRequest with type Immediate
Tool validations	<ul style="list-style-type: none"> * Step 2: Message: ResetResponse - status must be <i>Accepted</i> 	
Post condition	State is <i>Booting</i>	

Table 232. Reusable State: Booted

State	Booted	
System under test	Charging Station	
Description	This state will reset or power cycle the Charging Station, depending on the testcase. The charging station ends in a state where it is booted back up and is in idle mode.	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): N/a	
Main (Scenario)	Charging Station <i>Manual Action: Power cycle the Charging Station.</i> OR execute step 1 and 2, depending on the testcase.	CSMS
	2. The Charging Station responds with a ResetResponse with status Accepted	1. The OCTT sends a ResetRequest
	3. The Charging Station sends a BootNotificationRequest	4. The OCTT responds with a BootNotificationResponse with status Accepted
	5. The Charging Station notifies the CSMS about the current state of all connectors.	6. The OCTT responds accordingly.
	7 The Charging Station sends a SecurityEventNotificationRequest	8 The OCTT responds with a SecurityEventNotificationResponse
Tool validations	* Step 2: Message: ResetResponse - status Accepted * Step 5: Message: StatusNotificationRequest - connectorStatus Available - evsId not 0 - connectorId not 0 Message: NotifyEventRequest - eventData[0].trigger Delta - eventData[0].actualValue "Available" - eventData[0].component.name "Connector" - eventData[0].variable.name "AvailabilityState" * Step 7: Message: SecurityEventNotificationRequest - type must be <i>StartupOfTheDevice</i> OR <i>ResetOrReboot</i>	
Post condition	State is Booted	

Table 233. Reusable State: Reserved

State	Reserved	
System under test	Charging Station	
Description	This state will prepare the Charging Station, so that one of its EVSE becomes reserved.	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): N/a	
Main (Scenario)	Charging Station 2. The Charging Station responds with a ReserveNowResponse Note(s): - The OCTT expects that the Charging Station sets the availabilityState of the EVSE and corresponding connectors to Reserved. - Reporting the AvailabilityState of the EVSE component itself is optional.	CSMS 1. The OCTT sends a ReserveNowRequest with evselid is <Specified evselid (Configured evselid as a default)> idToken.idToken <Specified valid_idtoken_idtoken (Configured idToken as a default)> idToken.type <Specified valid_idtoken_type> 3. The Charging Station notifies the CSMS about the status change of the connector. 4. The OCTT responds accordingly.
Tool validations	<ul style="list-style-type: none"> * Step 2: Message: ReserveNowResponse - status must be Accepted * Step 3: Message: StatusNotificationRequest - evselid not 0 - connectorId not 0 - connectorStatus must be Reserved Message: NotifyEventRequest - eventData[0].trigger must be Delta - eventData[0].actualValue must be Reserved - eventData[0].component.name must be Connector - eventData[0].evse.id not 0 - eventData[0].evse.connectorId not 0 - eventData[0].variable.name must be AvailabilityState (Optional) Message: NotifyEventRequest - eventData[0].trigger must be Delta - eventData[0].actualValue must be Reserved - eventData[0].component.name must be EVSE - eventData[0].variable.name must be AvailabilityState 	
Post condition	State is Reserved	

Table 234. Reusable State: Unavailable

State	Unavailable	
System under test	Charging Station	
Description	This state will prepare the Charging Station, so that the Charging Station / EVSEs / connectors are set to AvailabilityState Unavailable.	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): N/a	
Main (Scenario)	Charging Station 2. The Charging Station responds with a ChangeAvailabilityResponse	CSMS 1. The OCTT sends a ChangeAvailabilityRequest with operationalStatus Inoperative evse.id <Specified evsId> evse.connectorId <Specified connectorId>
	3. The Charging Station notifies the CSMS about the current state of all connectors belonging to the specified component(s).	4. The OCTT responds accordingly.
Tool validations	* Step 2: Message ChangeAvailabilityResponse - status Accepted * Step 3: Message: StatusNotificationRequest - connectorStatus Unavailable - evsId <Specified evsId> - connectorId <Specified connectorId> Message: NotifyEventRequest - eventData[0].trigger Delta - eventData[0].actualValue "Unavailable" - eventData[0].component.name "ChargingStation" / EVSE / Connector - eventData[0].variable.name "AvailabilityState"	
Post condition	State is Reserved	

Table 235. Reusable State: ParkingBayOccupied

State	ParkingBayOccupied	
System under test	Charging Station	
Description	This state will prepare the Charging Station, so that the EV entered the parking bay. The execution of this State is optional . Because there may not be a parking bay occupancy sensor OR the Charging Station is being tested with a test plug or EV simulator.	
Before (Preparations)	<p>Configuration State: N/a</p> <p>Memory State: N/a</p> <p>Reusable State(s): N/a</p>	
Main (Scenario)	<p>Charging Station</p> <p><u>Manual Action:</u> Drive EV into parking bay.</p> <p><u>Note(s):</u> - This State is optional (Even when <i>TxStartPoint</i> contains <i>ParkingBayOccupancy</i>).</p>	<p>CSMS</p> <p>1. The Charging Station sends a <i>TransactionEventRequest</i></p> <p><u>Note(s):</u> - This step needs to be executed when <i>TxStartPoint</i> contains <i>ParkingBayOccupancy</i> AND the EV entered the parking bay.</p> <p>2. The OCTT responds with a <i>TransactionEventResponse</i></p>
Tool validations	<ul style="list-style-type: none"> * Step 1: Message: TransactionEventRequest - triggerReason must be <i>EVDetected</i> 	
Post condition	State is ParkingBayOccupied	

Table 236. Reusable State: EVConnectedPreSession

State	EVConnectedPreSession	
System under test	Charging Station	
Description	This state will prepare the Charging Station, so that the EV and EVSE are connected.	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): If State is NOT ParkingBayOccupied then execute Reusable State ParkingBayOccupied	
Main (Scenario)	Charging Station Manual Action: Connect the EV and EVSE. 1. The Charging Station notifies the CSMS about the status change of the connector. 3. The Charging Station sends a TransactionEventRequest Note(s): - This step needs to be executed when TxStartPoint contains EVConnected OR the transaction already started. So in the case TxStartPoint contains ParkingBayOccupancy OR Authorized	CSMS 2. The OCTT responds accordingly. 4. The OCTT responds with a TransactionEventResponse
Tool validations	* Step 1: Message: StatusNotificationRequest - evseld <configured evseld> - connectorId <configured connectorId> - connectorStatus must be Occupied Message: NotifyEventRequest - eventData[0].trigger must be <i>Delta</i> - eventData[0].actualValue must be <i>Occupied</i> - eventData[0].component.name must be <i>Connector</i> - eventData[0].variable.name must be <i>AvailabilityState</i> - evse.id <configured evseld> - connector.id <configured connectorId> * Step 3: Message: TransactionEventRequest - eventType started if TxStartPoint is EVConnected or PowerPathClosed and State is Authorized , else updated - triggerReason must be <i>CablePluggedIn</i> or <i>ChargingStateChanged</i> or <i>RemoteStart</i> - transactionInfo.chargingState must be EVConnected or SuspendedEVSE or Charging if State is Authorized - evse.id <configured evseld> - connector.id <configured connectorId>	
Post condition	State is EVConnectedPreSession	

Table 237. Reusable State: Authorized

State	Authorized	
System under test	Charging Station	
Description	<p>This state will prepare the Charging Station, so that the transaction is authorized. This can be done in two ways (The default way is configurable at OCTT. This will be used when the calling testcase does not define which one to use.):</p> <ul style="list-style-type: none"> A. Using local authorization B. Using a RequestStartTransactionRequest 	
Before (Preparations)	<p>Configuration State: N/a</p> <p>Memory State: N/a</p> <p>Reusable State(s): If State is NOT <i>ParkingBayOccupied</i> OR <i>EVConnectedPreSession</i>, then execute Reusable State <i>ParkingBayOccupied</i></p>	
Main A (Scenario)	Charging Station <i>Manual Action: Present idToken.</i> <ol style="list-style-type: none"> 1. The Charging Station sends an AuthorizeRequest <p>Note(s): - This step needs to be executed, unless (AuthEnabled is implemented with mutability <i>ReadOnly</i> AND the value is set to <i>false</i>) OR a start button as described at Use case C02 is used (This must be configured at the OCTT) OR the idToken is cached. In case the idToken is used for a reservation, sending the AuthorizeRequest message is optional.</p> 2. The OCTT responds with an AuthorizeResponse with idTokenInfo.status Accepted <ol style="list-style-type: none"> 3. The Charging Station sends a TransactionEventRequest <p>Note(s): - This step needs to be executed when TxStartPoint contains Authorized OR the transaction already started. So in the case TxStartPoint contains ParkingBayOccupancy or (EVConnected, in the case this testcase was initiated from state EVConnectedPreSession.)</p> 4. The OCTT responds with a TransactionEventResponse <p>Note(s): - The first TransactionEventRequest sent after authorization contains the idToken field, unless a Start button was used to start the transaction. In case there is an idToken used, the TransactionEventResponse of this request message contains idTokenInfo with status Accepted</p> 	
Tool validations	<ul style="list-style-type: none"> * Step 1: Message: AuthorizeRequest <ul style="list-style-type: none"> - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type> * Step 3: Message: TransactionEventRequest <ul style="list-style-type: none"> - triggerReason must be Authorized - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type> 	

State	Authorized	
Main B (Scenario)	<p>Charging Station</p> <p>2. The Charging Station responds with a RequestStartTransactionResponse</p>	<p>CSMS</p> <p>1. The OCTT sends a RequestStartTransactionRequest with idToken.idToken <Configured valid_idtoken_idtoken> idToken.type <Configured valid_idtoken_type> evsId <Configured evsId></p>
	<p>3. The Charging Station sends an AuthorizeRequest</p> <p><u>Note(s):</u> - This step needs to be executed when AuthCtrlr.AuthorizeRemoteStart is true, unless (AuthEnabled is implemented with mutability ReadOnly AND the value is set to false) OR the idToken is cached. In case the idToken is used for a reservation, sending the AuthorizeRequest message is optional.</p>	<p>4. The OCTT responds with an AuthorizeResponse with idTokenInfo.status Accepted</p>
	<p>5. The Charging Station sends a StatusNotificationRequest with: connectorStatus Occupied</p>	<p>6. The OCTT responds with a StatusNotificationResponse</p>
	<p>7. The Charging Station sends a TransactionEventRequest</p> <p><u>Note(s):</u> - This step needs to be executed when TxStartPoint contains Authorized OR the transaction already started. So in the case TxStartPoint contains ParkingBayOccupancy or (EVConnected, in the case this testcase was initiated from state EVConnectedPreSession.)</p>	<p>8. The OCTT responds with a TransactionEventResponse</p> <p><u>Note(s):</u> - The first TransactionEventRequest sent after authorization contains the idToken field. The TransactionEventResponse of this request message contains idTokenInfo with status Accepted</p>
Tool validations	<ul style="list-style-type: none"> * Step 2: <p>Message: RequestStartTransactionResponse</p> <ul style="list-style-type: none"> - status must be Accepted If the transaction has already been started, so if TxStartPoint contains ParkingBayOccupancy OR (<Configured TxStartPoint> contains EVConnected AND State pre reusable state execution was EVConnectedPreSession) then - transactionId must be <Provided transactionId in first TransactionEventRequest> <ul style="list-style-type: none"> * Step 3: <p>Message: AuthorizeRequest</p> <ul style="list-style-type: none"> - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type> <ul style="list-style-type: none"> * Step 5: <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - eventType Started if TxStartPoint is Authorized or PowerPathClosed and and State is EVConnectedPreSession, else updated - triggerReason must be RemoteStart - transactionInfo.remoteStartId must be present. - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type> 	
Post condition	State is Authorized	

Table 238. Reusable State: Authorized15118

State	Authorized15118	
System under test	Charging Station	
Description	<p>This state will prepare the Charging Station, so that the transaction is authorized. This can be done in two ways based on the value of the <i>Authorization Method</i> config variable:</p> <ul style="list-style-type: none"> A. EIM, using a valid id token B. PnG, plug and charge 	
Before (Preparations)	<p>Configuration State: N/a</p> <p>Memory State: N/a</p> <p>Reusable State(s): N/a</p>	
Main (Scenario)	<p>Charging Station</p> <p><u>Manual Action:</u> Present <i>idToken</i> if configured authorization method is <i>EIM</i></p> <p>Note(s): -The test case should be robust enough to also handle a GetCertificateStatusRequest and then expect the AuthorizeRequest.</p>	<p>CSMS</p> <p>2. The OCTT responds with an AuthorizeResponse with idTokenInfo.status Accepted</p>

Table 239. Reusable State: EnergyTransferStarted

State	EnergyTransferStarted	
System under test	Charging Station	
Description	This state will prepare the Charging Station, so that the Charging Station is transferring energy between the EV and EVSE.	
Before (Preparations)	<p>Configuration State: N/a</p> <p>Memory State: N/a</p> <p>Reusable State(s): If State is NOT Authorized then execute Reusable State Authorized If EVConnected is true, then proceed to part 2 Else proceed to part 1.</p>	
Main (Part 1) (Scenario)	Charging Station <u>Manual Action: Connect the EV and EVSE.</u> <p>1. The Charging Station notifies the CSMS about the status change of the connector.</p> <p>3. The Charging Station sends a TransactionEventRequest</p> <p><u>Note(s):</u> - This step needs to be executed when TxStartPoint contains EVConnected OR the transaction already started. So in the case TxStartPoint contains ParkingBayOccupancy OR Authorized</p>	CSMS <p>2. The OCTT responds accordingly.</p> <p>4. The OCTT responds with a TransactionEventResponse</p>
Tool validations	<ul style="list-style-type: none"> * Step 1: Message: StatusNotificationRequest - connectorStatus must be <i>Occupied</i> Message: NotifyEventRequest - eventData[0].trigger must be <i>Delta</i> - eventData[0].actualValue must be <i>Occupied</i> - eventData[0].component.name must be <i>Connector</i> - eventData[0].variable.name must be <i>AvailabilityState</i> * Step 3: Message: TransactionEventRequest - triggerReason must be <i>CablePluggedIn</i> - transactionInfo.chargingState must be <i>EVConnected</i> 	

State	EnergyTransferStarted	
Main (Part 2) (Scenario)	<p>Charging Station</p> <p>5. The Charging Station sends a TransactionEventRequest</p> <p>Note(s): - This step only needs to be executed when TxStartPoint contains DataSigned AND the transaction was not already started. So in the case TxStartPoint also contains ParkingBayOccupancy OR EVConnected OR Authorized</p>	<p>CSMS</p> <p>6. The OCTT responds with a TransactionEventResponse</p>
	<p>7. The Charging Station sends a TransactionEventRequest</p> <p>Note(s): - This step only needs to be executed when TxStartPoint contains PowerPathClosed AND the transaction was not already started. So in the case TxStartPoint also contains ParkingBayOccupancy OR EVConnected OR Authorized OR DataSigned</p>	<p>8. The OCTT responds with a TransactionEventResponse</p>
	<p>9. The Charging Station sends a TransactionEventRequest</p> <p>Note(s): - This step needs to be executed when TxStartPoint contains EnergyTransfer OR the transaction already started. So in the case TxStartPoint contains ParkingBayOccupancy OR EVConnected OR Authorized OR DataSigned OR PowerPathClosed</p>	<p>10. The OCTT responds with a TransactionEventResponse</p>
Tool validations	<ul style="list-style-type: none"> * Step 5: Message: TransactionEventRequest - triggerReason must be <i>SignedDataReceived</i> * Step 7: Message: TransactionEventRequest - triggerReason must be <i>ChargingStateChanged</i> - transactionInfo.chargingState must be <i>SuspendedEVSE</i> * Step 9: Message: TransactionEventRequest - triggerReason must be <i>ChargingStateChanged</i> - transactionInfo.chargingState must be <i>Charging</i> 	
Post condition	<p>State is <i>EnergyTransferStarted</i> EVConnected is <i>true</i></p>	

Table 240. Reusable State: EnergyTransferSuspended

State	EnergyTransferSuspended	
System under test	Charging Station	
Description	This state will prepare the Charging Station, so that it is in a state where the energy transfer is suspended by the EV.	
Prerequisite	N/a	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): If State is NOT <i>EnergyTransferStarted</i> then execute Reusable State <i>EnergyTransferStarted</i>	
Main (Scenario)	Charging Station <u>Notes(s):</u> The tool will wait for <Configured Transaction Duration> seconds <u>Manual Action:</u> The EV suspends the energy transfer.	CSMS 2. The OCTT responds with a TransactionEventResponse
Tool validations	* Step 1: Message: TransactionEventRequest - triggerReason must be <i>ChargingStateChanged</i> - transactionInfo.chargingState must be <i>EVConnected</i> OR - transactionInfo.chargingState must be <i>SuspendedEV</i> AND - transactionInfo.stoppedReason must be <i>StoppedByEV</i> - eventType must be <i>Ended</i> OR <i>Updated</i>	
Post condition	State is <i>EnergyTransferSuspended</i>	

Table 241. Reusable State: StopAuthorized

State	StopAuthorized							
System under test	Charging Station							
Description	<p>This state will prepare the Charging Station, so that it is in a state where the charging session is authorized to stop. This can be done in two ways (Configurable at OCTT):</p> <ul style="list-style-type: none"> A. Using local authorization B. Using a RequestStopTransactionRequest 							
Before (Preparations)	<p>Configuration State: N/a</p> <p>Memory State: N/a</p> <p>Reusable State(s): If State is NOT <i>EnergyTransferStarted</i> then execute Reusable State <i>EnergyTransferStarted</i></p> <p><u>Note:</u> The OCTT will wait a number of seconds equal to the configured <TransactionDuration>, before proceeding to the Main stage.</p>							
Main A (Scenario)	<p>Charging Station</p> <p><u>Notes(s):</u> The tool will wait for <Configured Transaction Duration> seconds</p> <p><u>Manual Action:</u> Present the same idToken as used to start the transaction.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="padding: 5px;">1. The Charging Station sends a TransactionEventRequest</td> <td style="padding: 5px;">2. The OCTT responds with a TransactionEventResponse With idTokenInfo.status is Accepted</td> </tr> <tr> <td colspan="2" style="padding: 5px;"><u>Note(s):</u> This step is optional</td> </tr> <tr> <td style="padding: 5px;">3. The Charging Station sends a TransactionEventRequest</td> <td style="padding: 5px;">4. The OCTT responds with a TransactionEventResponse With idTokenInfo.status is Accepted</td> </tr> </table>	1. The Charging Station sends a TransactionEventRequest	2. The OCTT responds with a TransactionEventResponse With idTokenInfo.status is Accepted	<u>Note(s):</u> This step is optional		3. The Charging Station sends a TransactionEventRequest	4. The OCTT responds with a TransactionEventResponse With idTokenInfo.status is Accepted	<p>CSMS</p>
1. The Charging Station sends a TransactionEventRequest	2. The OCTT responds with a TransactionEventResponse With idTokenInfo.status is Accepted							
<u>Note(s):</u> This step is optional								
3. The Charging Station sends a TransactionEventRequest	4. The OCTT responds with a TransactionEventResponse With idTokenInfo.status is Accepted							
Tool validations	<ul style="list-style-type: none"> * Step 1: Message: TransactionEventRequest <ul style="list-style-type: none"> - triggerReason must be StopAuthorized - idToken omit OR - idToken.idToken <Configured valid_idtoken_idtoken> AND - idToken.type <Configured valid_idtoken_type> * Step 3: Message: TransactionEventRequest <ul style="list-style-type: none"> - triggerReason must be ChargingStateChanged - transactionInfo.chargingState must be EVConnected - eventType must be Ended - transactionInfo.stoppedReason must be <i>Local</i> or omitted 							
Main B (Scenario)	<p>Charging Station</p> <p>2. The Charging Station responds with a RequestStopTransactionResponse</p> <p>3. The Charging Station sends a TransactionEventRequest</p>	<p>CSMS</p> <p>1. The OCTT sends a RequestStopTransactionRequest with transactionId <transactionId provided by the Charging Station in TransactionEventRequest></p> <p>4. The OCTT responds with a TransactionEventResponse</p>						
Tool validations	<ul style="list-style-type: none"> * Step 2: Message: RequestStopTransactionResponse <ul style="list-style-type: none"> - status must be Accepted * Step 3: Message: TransactionEventRequest <ul style="list-style-type: none"> - triggerReason must be RemoteStop 							

State	StopAuthorized
Post condition	State is <i>StopAuthorized</i>

Table 242. Reusable State: EVConnectedPostSession

State	EVConnectedPostSession	
System under test	Charging Station	
Description	This state will prepare the Charging Station, so that energy transfer has been stopped and the transaction is NOT authorized to resume energy transfer without re-authorization.	
Before (Preparations)	<p>Configuration State: N/a</p> <p>Memory State: N/a</p> <p>Reusable State(s): If State is NOT <i>StopAuthorized</i> then execute Reusable State StopAuthorized</p>	
Main (Scenario)	<p>Charging Station</p> <p>1. The Charging Station sends a TransactionEventRequest</p> <p>Note(s): - This step needs to be executed when the transaction has NOT been ended already. So in the case TxStopPoint contains Authorized OR PowerPathClosed</p> <p>3. The Charging Station sends a TransactionEventRequest</p> <p>Note(s): - This step only needs to be executed when TxStopPoint contains DataSigned AND the transaction has NOT been ended already. So in the case TxStopPoint contains Authorized OR EnergyTransfer OR PowerPathClosed</p>	<p>CSMS</p> <p>2. The OCTT responds with a TransactionEventResponse</p> <p>4. The OCTT responds with a TransactionEventResponse</p>
Tool validations	<ul style="list-style-type: none"> * Step 1: Message: TransactionEventRequest - triggerReason must be <i>ChargingStateChanged</i> - transactionInfo.chargingState must be <i>EVConnected</i> * Step 3: Message: TransactionEventRequest - triggerReason must be <i>SignedDataReceived</i> 	
Post condition	State is <i>EVConnectedPostSession</i>	

Table 243. Reusable State: EVDisconnected

State	EVDisconnected	
System under test	Charging Station	
Description	This state will prepare the Charging Station, so that the EV and EVSE are disconnected, after the charging session is authorized to stop.	
Before (Preparations)	<p>Configuration State: N/a</p> <p>Memory State: N/a</p> <p>Reusable State(s): If State is NOT <i>EVConnectedPostSession</i> then execute Reusable State EVConnectedPostSession</p>	
Main (Scenario)	<p>Charging Station</p> <p><u>Manual Action:</u> Disconnect the EV and EVSE.</p> <p>1. The Charging Station notifies the CSMS about the status change of the connector.</p> <p>3. The Charging Station sends a TransactionEventRequest</p> <p><u>Note(s):</u> - This step needs to be executed when the transaction has NOT been ended already. So in the case TxStopPoint contains Authorized OR EnergyTransfer OR PowerPathClosed OR DataSigned</p>	<p>CSMS</p> <p>2. The OCTT responds accordingly.</p> <p>4. The OCTT responds with a TransactionEventResponse</p>
Tool validations	<ul style="list-style-type: none"> * Step 1: Message: StatusNotificationRequest <ul style="list-style-type: none"> - connectorStatus must be <i>Available</i> - evseld must be <configured evseld> - connectorId must be <configured connectorId> Message: NotifyEventRequest <ul style="list-style-type: none"> - eventData[0].trigger must be <i>Delta</i> - eventData[0].actualValue must be <i>Available</i> - eventData[0].component.name must be <i>Connector</i> - eventData[0].variable.name must be <i>AvailabilityState</i> - eventData[0].component.evse.id must be <configured evseld> - eventData[0].component.evse.connectorId must be <configured connectorId> * Step 3: Message: TransactionEventRequest <ul style="list-style-type: none"> - triggerReason must be <i>EVCommunicationLost</i> - transactionInfo.chargingState must be <i>Idle</i> 	
Post condition	State is <i>EVDisconnected</i>	

Table 244. Reusable State: ParkingBayUnoccupied

State	ParkingBayUnoccupied	
System under test	Charging Station	
Description	This state will prepare the Charging Station, so that the EV left the parking bay, after a charging session has taken place.	
Before (Preparations)	<p>Configuration State: N/a</p> <p>Memory State: N/a</p> <p>Reusable State(s): If State is NOT EVDisconnected then execute Reusable State EVDisconnected</p>	
Main (Scenario)	Charging Station <u>Manual Action:</u> <i>Drive EV out of parking bay.</i> 1. The Charging Station sends a TransactionEventRequest <u>Note(s):</u> - This step needs to be executed when TxStopPoint contains <i>ParkingBayOccupancy</i> AND the transaction has NOT been ended already. So in the case TxStopPoint contains <i>Authorized</i> OR <i>EnergyTransfer</i> OR <i>PowerPathClosed</i> OR <i>DataSigned</i> OR <i>EVConnected</i> .	CSMS 2. The OCTT responds with a TransactionEventResponse
Tool validations	<ul style="list-style-type: none"> * Step 1: Message: TransactionEventRequest - triggerReason must be <i>EVDeparted</i> - If the OCTT is configured to stop transactions using a RequestStopTransactionRequest message then transactionInfo.stoppedReason must be <i>Remote</i> Else transactionInfo.stoppedReason must be <i>Local</i> - eventType must be <i>Ended</i> 	
Post condition	State is <i>ParkingBayUnoccupied</i>	

Table 245. Reusable State: StartOfflineTransaction

State	StartOfflineTransaction	
System under test	Charging Station	
Description	This state will start a transaction while the Charging Station is offline.	
Prerequisite		
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Scenario)	Charging Station	CSMS
	1. The OCTT closes the WebSocket connection AND does not accept a reconnect.	
	<u>Manual Action:</u> Drive EV into parking bay.	
	<u>Manual Action:</u> Present idToken.	
	<u>Manual Action:</u> Connect the EV and EVSE.	
Tool validations	2. The OCTT accepts reconnection attempt from the Charging Station.	
	N/a	
Post condition	N/a	

Table 246. Reusable State: RenegotiateChargingLimits

State	RenegotiateChargingLimits	
System under test	Charging Station	
Description	...	
Prerequisite		
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Scenario)	Charging Station	CSMS
	1. The Charging Station sends a NotifyEVChargingNeedsRequest with evseld <Configured evseld>	2. The OCTT responds with a NotifyEVChargingNeedsResponse with status Accepted
	4. The Charging Station responds with a SetChargingProfileResponse with status Accepted	3. The OCTT sends a SetChargingProfileRequest with chargingProfile : .chargingProfilePurpose <i>TxProfile</i> .transactionId <Provided transactionId from before> chargingProfile.chargingSchedule[0] : .duration 300 .chargingRateUnit <Configured chargingRateUnit> Note: If <Configured chargingRateUnit> is W, then the <i>limit</i> field will be multiplied by 1000. .chargingSchedulePeriod[0].startPeriod 0 If <Configured chargingRateUnit> is W: .chargingSchedulePeriod[0].limit 10000 else: .chargingSchedulePeriod[0].limit 10
	5. The Charging Station sends a NotifyEVChargingScheduleRequest with evseld <Configured evseld>	6. The OCTT responds with a NotifyEVChargingScheduleResponse with status Accepted
	<i>Note: This step is optional. The Charging Station will only send it when the EV returns a charging profile.</i>	
Tool validations	7. The Charging Station sends a TransactionEventRequest	8. The OCTT responds with a TransactionEventResponse
	* Step 7: Message: TransactionEventRequest - triggerReason must be <i>ChargingStateChanged</i> - TransactionInfo.chargingState must be <i>Charging</i>	
Post condition	N/a	

Table 247. Reusable State: GetInstalledCertificates

State	GetInstalledCertificates	
System under test	Charging Station	
Description	The hashData from installed certificates of the specified type will be retrieved from the Charging Station	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Scenario)	Charging Station	CSMS
	2. The Charging Station responds with a GetInstalledCertificateIdsResponse	1. The OCTT sends a GetInstalledCertificateIdsRequest With certificateType is <Specified certificateType>
Tool validations	* Step 2: Message: GetInstalledCertificateIdsResponse - status must be Accepted - certificateHashDataChain must contain an entry with following values: <i>Note: Order does not matter.</i> - certificateHashDataChain[0].certificateType is <Specified certificateType> - certificateHashDataChain[0].certificateHashData contains <HashData from the configured certificate of the specified certificateType>	
Post condition	Certificate of the specified certificateType is retrieved from the Charging Station.	

2.19. Memory states

Table 248. Memory State: TransactionEventsInQueueEnded

State	TransactionEventsInQueueEnded	
System under test	Charging Station	
Description	This state will prepare the Charging Station, so that there will be TransactionEventRequests stored in its queue from an ended Transaction.	
Before (Preparations)	<p>Configuration State: OfflineTxForUnknownIdEnabled is <i>true</i> (If implemented)</p> <p>Memory State: <i>IdTokenCached</i> for <Configured valid IdToken fields> (If implemented) <i>IdTokenLocalAuthList</i> for <Configured valid IdToken fields> (If implemented)</p> <p>Reusable State(s): N/a</p>	
Main (Scenario)	Charging Station	CSMS
	<p>1. The OCTT closes the WebSocket connection AND does not accept a reconnect.</p> <p><u>Manual Action:</u> <i>Drive EV into parking bay.</i></p> <p><u>Manual Action:</u> <i>Connect the EV and EVSE.</i></p> <p><u>Manual Action:</u> <i>Present idToken.</i></p> <p><u>Manual Action:</u> <i>Present the same idToken as used to start the transaction.</i></p> <p><u>Manual Action:</u> <i>Disconnect the EV and EVSE.</i></p> <p><u>Manual Action:</u> <i>Drive EV out of parking bay.</i></p> <p>2. The OCTT accepts reconnection attempt from the Charging Station.</p>	
Tool validations	N/a	
Post condition	TransactionEventRequest messages are stored in the queue of the Charging Station.	

Table 249. Memory State: CertificateInstalled

State	CertificateInstalled	
System under test	Charging Station	
Description	A pre configured certificate of the specified certificateType will be installed.	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): N/a	
Main (Scenario)	Charging Station 2. The Charging Station responds with a InstallCertificateResponse	CSMS 1. The OCTT sends a InstallCertificateRequest with certificateType is <Specified certificateType> certificate is <Corresponding certificate>
Tool validations	* Step 2: Message: InstallCertificateResponse - status must be Accepted	
Post condition	Certificate of the specified certificateType is stored at the Charging Station.	

Table 250. Memory State: *IdTokenCached*

State	IdTokenCached	
System under test	Charging Station	
Description	An idToken is stored in the Authorization Cache of the Charging Station.	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): N/a	
Main (Scenario)	Charging Station	CSMS
	1. Execute Reusable State <i>ParkingBayoccupied</i> 2. Execute Reusable State <i>EVConnectedPreSession</i> 3. Execute Reusable State <i>Authorized</i>	
Main A (Scenario)	Charging Station	CSMS
	<u>Note(s):</u> In case idToken is Accepted 4. Execute Reusable State <i>ParkingBayUnoccupied</i>	
Tool validations	N/a	
Main B (Scenario)	Charging Station	CSMS
	<u>Note(s):</u> In case idToken is not Accepted <u>Manual Action:</u> Unplug Cable 4. The Charging Station sends a TransactionEventRequest 5. The OCTT responds with a TransactionEventResponse 6. The Charging Station sends a StatusNotificationRequest or NotifyEventRequest 7. The OCTT responds with a StatusNotificationResponse or NotifyEventResponse 8. Execute Reusable State <i>ParkingBayUnoccupied</i>	
Tool validations	* Step 1: Message: TransactionEventRequest - triggerReason must be <i>EVConnectionLost</i> - transactionInfo.chargingState must be <i>Idle</i> * Step 3: Message: StatusNotificationRequest - connectorStatus must be <i>Available</i> Message: NotifyEventRequest - trigger must be <i>Delta</i> - actualValue must be <i>Available</i> - component.name must be <i>Connector</i> - variable.name must be <i>AvailabilityState</i>	
Post condition	N/a	

Table 251. Memory State: *IdTokenLocalAuthList*

State	IdTokenLocalAuthList	
System under test	Charging Station	
Description	An valid idToken is stored in the Local Authorization List of the Charging Station.	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Scenario)	Charging Station	CSMS
	2. The Charging Station responds with a SendLocalListResponse	1. The OCTT sends a SendLocalListRequest with updateType Full localAuthorizationList[0].idToken.idToken <i><Configured valid_idtoken_idtoken></i> localAuthorizationList[0].idToken.type <i><Configured valid_idtoken_type></i>
Tool validations	* Step 2: (Message: SendLocalListResponse) status is Accepted	
Post condition	N/a	

Table 252. Memory State: SetChargingProfile

State	SetChargingProfile	
System under test	Charging Station	
Description	This will store a Charging Profile at the Charging Station.	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): N/a	
Main (Scenario)	Charging Station 2. The Charging Station responds with a SetChargingProfileResponse	CSMS 1. The OCTT sends a SetChargingProfileRequest with chargingProfile <Provided chargingProfile>
Tool validations	* Step 2: (Message: SetChargingProfileResponse) status is Accepted	
Post condition	N/a	

Table 253. Memory State: RenewChargingStationCertificate

State	RenewChargingStationCertificate	
System under test	Charging Station	
Description	The ChargingStationCertificate is renewed using A02/A03	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): N/a	
Main (Test scenario)	Charging Station 2. The Charging Station responds with a TriggerMessageResponse 3 The Charging Station sends a SignCertificateRequest	CSMS 1. The OCTT sends a TriggerMessageRequest With requestedMessage <i>SignChargingStationCertificate</i>
	6. The Charging Station responds with a CertificateSignedResponse	4. The OCTT responds with a SignCertificateResponse With status Accepted 5. The OCTT sends a CertificateSignedRequest With certificateChain < <i>Certificate generated from the received CSR from step 3 and signed by the provided CSMS Root certificate</i> > certificateType <i>ChargingStationCertificate</i>
Tool validations	<ul style="list-style-type: none"> * Step 2: Message: TriggerMessageResponse - status must be Accepted * Step 3: Message: SignCertificateRequest - csr must contain <<i>An CSR that meets the following requirements: When using RSA or DSA the key must be at least 2048 bits long. and when using elliptic curve cryptography the key must be at least 224 bits long. The received CSR must be transmitted as described in RFC 2986 and then encoded in Privacy-Enhanced Mail (PEM) format.</i>> * Step 6: Message: CertificateSignedResponse - status must be Accepted <p>Post scenario validations: N/a</p>	

Table 254. Memory State: RenewV2GChargingStationCertificate

State	RenewV2GChargingStationCertificate	
System under test	Charging Station	
Description	The V2G ChargingStationCertificate is renewed using A02/A03	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): N/a	
Main (Test scenario)	Charging Station 2. The Charging Station responds with a TriggerMessageResponse 3 The Charging Station sends a SignCertificateRequest	CSMS 1. The OCTT sends a TriggerMessageRequest With requestedMessage SignV2GCertificate 4. The OCTT responds with a SignCertificateResponse With status Accepted
	 6. The Charging Station responds with a CertificateSignedResponse	 5. The OCTT sends a CertificateSignedRequest With certificateChain <Certificate generated from the received CSR from step 3 and signed by the V2GRoot OR SubCA certificate from the provided V2G certificate chain> certificateType V2GCertificate
Tool validations	<ul style="list-style-type: none"> * Step 2: Message: TriggerMessageResponse - status must be <i>Accepted</i> * Step 3: Message: SignCertificateRequest - csr must contain <i><An CSR that meets the following requirements: The key must be at least 224 bits long. The received CSR must be transmitted as described in RFC 2986 and then encoded in Privacy-Enhanced Mail (PEM) format.></i> * Step 6: Message: CertificateSignedResponse - status must be <i>Accepted</i> <p>Post scenario validations: N/a</p>	

3. Test Cases Charging Station Management System

3.1. General pre/post conditions & tool validations

General conditions/validations are overruled by testcase specific conditions/validations, unless specifically stated otherwise.

General pre conditions:

The following pre conditions apply to all test cases, unless explicitly mentioned otherwise.

- The Configuration variable **TxCtrlr.TxStartPoint** is "*EVConnected,Authorized*"
- The Configuration variable **TxCtrlr.TxStopPoint** is "*EVConnected*"
- The Configuration variable **AuthCtrlr.AuthEnabled** is *true*
- The Configuration variable **AuthCtrlr.AuthorizeRemoteStart** is *false*
- The Configuration variable **AdditionalRootCertificateCheck** is *false*
- The Configuration variable **AllowNewSessionsPendingFirmwareUpdate** is *false*
- The Configuration variable **AlignedDataSendDuringIdle** is *false*

General tool rules/validations:

- The list of ChargingSchedulePeriod elements in a chargingSchedule SHALL be ordered by increasing values of ChargingSchedulePeriod.startPeriod. This means the list is in chronological order.
- The CSMS SHALL NOT set phaseToUse in a SetChargingProfileRequest when numberPhases is other than 1.

3.2. A Security

Table 255. Test Case Id: TC_A_01_CSMS

Test case name	Basic Authentication - Valid username/password combination	
Test case Id	TC_A_01_CSMS	
Use case Id(s)	A00, B01	
Requirement(s)	A00.FR.204, B01.FR.02	
System under test	CSMS	
Description	The Charging Station uses Basic authentication to authenticate itself to the CSMS, when using security profile 1 or 2.	
Purpose	To verify whether the CSMS is able to validate the (valid) Basic authentication credentials provided by the Charging Station at the connection request.	
Prerequisite(s)	The CSMS supports security profile 1 and/or 2	
Before (Preparations)	<p>Configuration State: The CSMS must have a password configured that equals the configured BasicAuthPassword at the OCTT.</p> <p>Memory State: N/a</p> <p>Reusable State(s): N/a</p>	
Main (Test scenario)	<p>Charging Station</p> <p>1. The OCTT sends a HTTP upgrade request with an Authorization header, containing a username/password combination.</p> <p>Note(s): - The Authorization header is formatted as follows: AUTHORIZATION: Basic <Base64 encoded(<Configured ChargingStationId>:<Configured BasicAuthPassword>)></p> <p>3. The OCTT sends a BootNotificationRequest</p> <p>5. The OCTT notifies the CSMS about the current state of all connectors.</p>	<p>CSMS</p> <p>2. The CSMS validates the username/password combination AND upgrades the connection to a (secured) WebSocket connection.</p> <p>4. The CSMS responds with a BootNotificationResponse</p> <p>6. The CSMS responds accordingly.</p>
Tool validations	<p>* Step 4: Message: BootNotificationResponse - status must be Accepted</p> <p>Post scenario validations: N/a</p>	

Table 256. Test Case Id: TC_A_02_CSMS

Test case name	Basic Authentication - Username does not equal ChargingStationId	
Test case Id	TC_A_02_CSMS	
Use case Id(s)	A00	
Requirement(s)	A00.FR.204	
System under test	CSMS	
Description	The Charging Station uses Basic authentication to authenticate itself to the CSMS, when using security profile 1 or 2.	
Purpose	To verify whether the CSMS is able to validate the (invalid) Basic authentication credentials provided by the Charging Station at the connection request.	
Prerequisite(s)	The CSMS supports security profile 1 and/or 2	
Before (Preparations)	<p>Configuration State: N/a</p> <p>Memory State: N/a</p> <p>Reusable State(s): N/a</p>	
Main (Test scenario)	<p>Charging Station</p> <p>1. The OCTT sends a HTTP upgrade request with an Authorization header, containing a username/password combination.</p> <p>Note(s): - The Authorization header is formatted as follows: AUTHORIZATION: Basic <Base64 encoded(<Configured ChargingStationId> + Invalid:<Configured basicAuthPassword>)></p>	<p>CSMS</p> <p>2. The CSMS validates the username/password combination AND rejects the connection upgrade request.</p>
Tool validations	<p>N/a</p> <p>Post scenario validations: N/a</p>	

Table 257. Test Case Id: TC_A_03_CSMS

Test case name	Basic Authentication - Invalid password	
Test case Id	TC_A_03_CSMS	
Use case Id(s)	A00	
Requirement(s)	A00.FR.204	
System under test	CSMS	
Description	The Charging Station uses Basic authentication to authenticate itself to the CSMS, when using security profile 1 or 2.	
Purpose	To verify whether the CSMS is able to validate the (invalid) Basic authentication credentials provided by the Charging Station at the connection request.	
Prerequisite(s)	The CSMS supports security profile 1 and/or 2	
Before (Preparations)	<p>Configuration State: N/a</p> <p>Memory State: N/a</p> <p>Reusable State(s): N/a</p>	
Main (Test scenario)	<p>Charging Station</p> <p>1. The OCTT sends a HTTP upgrade request with an Authorization header, containing a username/password combination.</p> <p>Note(s): - The Authorization header is formatted as follows: AUTHORIZATION: Basic <Base64 encoded(<Configured ChargingStationId>:<randomly chosen identifierString with a sufficiently high entropy, consisting of minimum 16 and maximum 40 characters (alpha-numeric characters and the special characters allowed by identifierString)>)</p>	<p>CSMS</p> <p>2. The CSMS validates the username/password combination AND rejects the connection upgrade request.</p>
Tool validations	<p>N/a</p> <p>Post scenario validations: N/a</p>	

Table 258. Test Case Id: TC_A_04_CSMS

Test case name	TLS - server-side certificate - Valid certificate
Test case Id	TC_A_04_CSMS
Use case Id(s)	A00
Requirement(s)	A00.FR.306,A00.FR.307,A00.FR.312,A00.FR.318,A00.FR.321,A00.FR.502,A00.FR.503,A00.FR.507,A00.FR.508,A00.FR.510
System under test	CSMS
Description	The CSMS uses a server-side certificate to identify itself to the Charging Station, when using security profile 2 or 3.
Purpose	To verify whether the CSMS is able to provide a valid server certificate and setup a secured WebSocket connection.
Prerequisite(s)	The CSMS supports security profile 2 and/or 3
Before (Preparations)	<p>Configuration State: N/a</p> <p>Memory State: N/a</p> <p>Reusable State(s): N/a</p>

Test case name	TLS - server-side certificate - Valid certificate	
Main (Test scenario)	Charging Station 1. The OCTT terminates the connection and initiates a TLS handshake and sends a Client Hello to the CSMS. 3. The OCTT performs the following actions: Send client certificate Client Key Exchange Certificate verify Change Cipher Spec Finished <u>Note(s):</u> - <i>The client certificate is only sent when the CSMS uses security profile 3.</i>	CSMS 2. The CSMS responds with a Server Hello With the <Configured server certificate> 4. The CSMS performs the following actions: Change Cipher Spec Finished
	5. The OCTT sends a HTTP upgrade request to the CSMS <u>Note(s):</u> - <i>The HTTP request only contains a username/password combination when the CSMS uses security profile 2.</i>	6. The CSMS upgrades the connection to a (secured) WebSocket connection.
	7. The OCTT sends a BootNotificationRequest with reason PowerUp chargingStation.model <Configured model> chargingStation.vendorName <Configured vendorName>	8. The CSMS responds with a BootNotificationResponse
	9. The OCTT notifies the CSMS about the current state of all connectors. Message: StatusNotificationRequest - connectorStatus Available Message: NotifyEventRequest - trigger Delta - actualValue "Available" - component.name "Connector" - variable.name "AvailabilityState"	10. The CSMS responds accordingly.

Test case name	TLS - server-side certificate - Valid certificate
Tool validations	<p>* Step 3:</p> <p>The OCTT validates the following before finishing the TLS handshake:</p> <ul style="list-style-type: none"> - The CSMS must use TLS version 1.2 or above <p>At least the following set of cipher suites must be supported:</p> <p>TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 AND TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 AND TLS_RSA_WITH_AES_128_GCM_SHA256 AND TLS_RSA_WITH_AES_256_GCM_SHA384</p> <ul style="list-style-type: none"> - When using RSA or DSA the key must be at least 2048 bits long. <p>and when using elliptic curve cryptography the key must be at least 224 bits long.</p> <ul style="list-style-type: none"> - The received server side certificate must be transmitted in the X.509 format encoded in Privacy-Enhanced Mail (PEM) format. - The certificate must include a serial number. - The subject field of the certificate must contain a commonName RDN which consists of the FQDN of the endpoint of the server. <p><i>NOTE: If one of the above validations fails, the OCTT can still proceed with the next steps of the testcase (if it is able to), but the testcase will FAIL and the OCTT reports why it failed.</i></p> <p>* Step 8:</p> <p>Message: BootNotificationResponse with status Accepted</p>
	Post scenario validations: N/a

Table 259. Test Case Id: TC_A_06_CSMS

Test case name	TLS - server-side certificate - TLS version too low	
Test case Id	TC_A_06_CSMS	
Use case Id(s)	A00	
Requirement(s)	A00.FR.314,A00.FR.315,A00.FR.409,A00.FR.416,A00.FR.417,A00.FR.418	
System under test	CSMS	
Description	The CSMS uses a server-side certificate to identify itself to the Charging Station, when using security profile 2 or 3.	
Purpose	To verify whether the CSMS is able to terminate the connection when it notices the used TLS version is lower than 1.2.	
Prerequisite(s)	The CSMS supports security profile 2 and/or 3	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): N/a	
Main (Test scenario)	Charging Station 1. The OCTT terminates the connection and initiates a TLS handshake with a TLS version lower than 1.2 and sends a Client Hello to the CSMS. 3. The OCTT initiates a TLS handshake with TLS version 1.2 or higher and sends a Client Hello to the CSMS. 5. The OCTT performs the following actions: Send client certificate Client Key Exchange Certificate verify Change Cipher Spec Finished <u>Note(s):</u> - The client certificate is only sent when the CSMS uses security profile 3.	CSMS 2. The CSMS notices that the TLS version is lower than 1.2 and terminates the connection. 4. The CSMS responds with a Server Hello With the <Configured server certificate> 6. The CSMS performs the following actions: Change Cipher Spec Finished
	 7. The OCTT sends a HTTP upgrade request to the CSMS <u>Note(s):</u> - The HTTP request only contains a username/password combination when the CSMS uses security profile 2.	 8. The CSMS upgrades the connection to a (secured) WebSocket connection.
	 9. The OCTT sends a BootNotificationRequest with reason PowerUp chargingStation.model <Configured model> chargingStation.vendorName <Configured vendorName>	 10. The CSMS responds with a BootNotificationResponse

Test case name	TLS - server-side certificate - TLS version too low	
	<p>11. The OCTT notifies the CSMS about the current state of all connectors.</p> <p>Message: StatusNotificationRequest - connectorStatus Available</p> <p>Message: NotifyEventRequest - trigger Delta - actualValue "Available" - component.name "Connector" - variable.name "AvailabilityState"</p>	<p>12. The CSMS responds accordingly.</p>
Tool validations	<p>* Step 10:</p> <p>Message: BootNotificationResponse - status Accepted</p> <p>Post scenario validations: N/a</p>	

Table 260. Test Case Id: TC_A_07_CSMS

Test case name	TLS - Client-side certificate - valid certificate	
Test case Id	TC_A_07_CSMS	
Use case Id(s)	A00	
Requirement(s)	A00.FR.409,A00.FR.410,A00.FR.415,A00.FR.416,A00.FR.421	
System under test	CSMS	
Description	The Charging Station uses a client-side certificate to identify itself to the CSMS, when using security profile 3.	
Purpose	To verify whether the CSMS is able to receive a client certificate provided by a Charging Station and setup a secured WebSocket connection.	
Prerequisite(s)	The CSMS supports security profile 3	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): N/a	
Main (Test scenario)	Charging Station 1. The OCTT terminates the connection and initiates a TLS handshake and sends a Client Hello to the CSMS. 3. The OCTT performs the following actions: Send <Configured client certificate> Client Key Exchange Certificate verify Change Cipher Spec Finished 5. The OCTT sends a HTTP upgrade request to the CSMS	CSMS 2. The CSMS responds with a Server Hello With the <Configured server certificate> 4. The CSMS performs the following actions: Change Cipher Spec Finished 6. The CSMS upgrades the connection to a (secured) WebSocket connection.
	7. The OCTT sends a BootNotificationRequest with reason PowerUp chargingStation.model <Configured model> chargingStation.vendorName <Configured vendorName> 9. The OCTT notifies the CSMS about the current state of all connectors. Message: StatusNotificationRequest - connectorStatus Available Message: NotifyEventRequest - trigger Delta - actualValue "Available" - component.name "Connector" - variable.name "AvailabilityState"	8. The CSMS responds with a BootNotificationResponse 10. The CSMS responds accordingly.

Test case name	TLS - Client-side certificate - valid certificate
Tool validations	<p>* Step 3:</p> <p>The OCTT validates the following before finishing the TLS handshake:</p> <ul style="list-style-type: none"> - The CSMS must use TLS version 1.2 or above <p>At least the following set of cipher suites must be supported:</p> <p>TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 AND TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 AND TLS_RSA_WITH_AES_128_GCM_SHA256 AND TLS_RSA_WITH_AES_256_GCM_SHA384</p> <p>* Step 8:</p> <p>Message: BootNotificationResponse with status Accepted</p>
	<p>Post scenario validations: N/a</p>

Table 261. Test Case Id: TC_A_08_CSMS

Test case name	TLS - Client-side certificate - Invalid certificate	
Test case Id	TC_A_08_CSMS	
Use case Id(s)	A00	
Requirement(s)	A00.FR.405,A00.FR.407,A00.FR.409,A00.FR.410	
System under test	CSMS	
Description	The Charging Station uses a client-side certificate to identify itself to the CSMS, when using security profile 3.	
Purpose	To verify whether the CSMS is able to terminate the connection when the received client certificate is invalid.	
Prerequisite(s)	<ul style="list-style-type: none"> - The CSMS supports security profile 3 - This testcase can be executed multiple times, using different kinds of invalid certificates: Unknown certificate expired certificate certificate with commonName that does not equal the serial number of the Charging Station.	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	1. The OCTT initiates a TLS handshake and sends a Client Hello to the CSMS.	2. The CSMS responds with a Server Hello With a server certificate
	3. The OCTT performs the following actions: Send <Configured invalid client certificate> Client Key Exchange Certificate verify Change Cipher Spec Finished	4. The CSMS deems the client certificate invalid and terminates the connection.
	5. The OCTT initiates a TLS handshake and sends a Client Hello to the CSMS.	6. The CSMS responds with a Server Hello With a server certificate
	7. The OCTT performs the following actions: Send <Configured client certificate> Client Key Exchange Certificate verify Change Cipher Spec Finished	8. The CSMS performs the following actions: Change Cipher Spec Finished
	9. The OCTT sends a HTTP upgrade request to the CSMS	10. The CSMS upgrades the connection to a (secured) WebSocket connection.
	11. The OCTT sends a BootNotificationRequest with reason PowerUp chargingStation.model <Configured model> chargingStation.vendorName <Configured vendorName>	12. The CSMS responds with a BootNotificationResponse

Test case name	TLS - Client-side certificate - Invalid certificate	
	<p>13. The OCTT notifies the CSMS about the current state of all connectors.</p> <p>Message: StatusNotificationRequest - connectorStatus <i>Available</i></p> <p>Message: NotifyEventRequest - trigger <i>Delta</i> - actualValue "Available" - component.name "Connector" - variable.name "AvailabilityState"</p>	<p>14. The CSMS responds accordingly.</p>
Tool validations	<p>* Step 12:</p> <p>Message: BootNotificationResponse with status Accepted</p> <p>Post scenario validations: N/a</p>	

Table 262. Test Case Id: TC_A_09_CSMS

Test case name	Update Charging Station Password for HTTP Basic Authentication - Accepted	
Test case Id	TC_A_09_CSMS	
Use case Id(s)	A01	
Requirement(s)	A01.FR.02, A01.FR.03	
System under test	CSMS	
Description	This test case defines how to use the BasicAuthPassword, the password used to authenticate Charging Stations in security profile 1 (Basic Authentication) and security profile 2 (TLS with Basic Authentication)	
Purpose	To verify if the CSMS is able to successfully set the new BasicAuthPassword and only accepts the new credentials as described at the OCPP specification.	
Prerequisite(s)	The CSMS supports security profile 1 and/or 2	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The OCTT responds with a SetVariablesResponse with status Accepted	<p>1. The CSMS sends a SetVariablesRequest with: setVariableData[1]:</p> <ul style="list-style-type: none"> - variable.name = "BasicAuthPassword" - component.name = "SecurityCtrlr" - attributeValue = "<NewPassword>"
	3. The OCTT sends a HTTP upgrade request with an Authorization header, containing a username/password combination (with the new <i>BasicAuthPassword</i>). <u>Note(s):</u> - The Authorization header is formatted as follows: AUTHORIZATION: Basic <Base64 encoded(<Configured ChargingStationId>:<NEW BasicAuthPassword>)>	4. The CSMS validates the username/password combination AND upgrades the connection to a (secured) WebSocket connection.
	5. The OCTT sends a BootNotificationRequest	6. The CSMS responds with a BootNotificationResponse
	7. The OCTT notifies the CSMS about the current state of all connectors.	8. The CSMS responds accordingly.
Tool validations	<ul style="list-style-type: none"> * Step 1: Message: SetVariableRequest - variable.name = "BasicAuthPassword" - component.name = "SecurityCtrlr" * Step 6: Message: BootNotificationResponse - status must be Accepted 	
	Post scenario validations: N/a	

Table 263. Test Case Id: TC_A_10_CSMS

Test case name	Update Charging Station Password for HTTP Basic Authentication - Rejected	
Test case Id	TC_A_10_CSMS	
Use case Id(s)	A01	
Requirement(s)	A01.FR.02, A01.FR.04, A01.FR.05	
System under test	CSMS	
Description	This test case defines how to use the BasicAuthPassword, the password used to authenticate Charging Stations in security profile 1 (Basic Authentication) and security profile 2 (TLS with Basic Authentication)	
Purpose	To verify if the CSMS keeps accepting the old credentials and keeps communication when the new BasicAuthPassword is rejected as described at the OCPP specification.	
Prerequisite(s)	The CSMS supports security profile 1 and/or 2	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The OCTT responds with a SetVariablesResponse with status Rejected	1. The CSMS sends a SetVariablesRequest with: setVariableData[1]: - variable.name = "BasicAuthPassword" - component.name = "SecurityCtrlr" - attributeValue = "<NewPassword>"
	3. The OCTT sends a HTTP upgrade request with an Authorization header, containing a username/password combination (with the old <i>BasicAuthPassword</i>). <u>Note(s):</u> - The Authorization header is formatted as follows: AUTHORIZATION: Basic <Base64 encoded(<Configured ChargingStationId>:<OLD Configured BasicAuthPassword>)>	4. The CSMS validates the username/password combination AND upgrades the connection to a (secured) WebSocket connection.
	5. The OCTT sends a BootNotificationRequest	6. The CSMS responds with a BootNotificationResponse
	7. The OCTT notifies the CSMS about the current state of all connectors.	8. The CSMS responds accordingly.
Tool validations	<ul style="list-style-type: none"> * Step 1: Message: SetVariableRequest - variable.name = "BasicAuthPassword" - component.name = "SecurityCtrlr" * Step 6: Message: BootNotificationResponse - status must be Accepted 	
	Post scenario validations: N/a	

Table 264. Test Case Id: TC_A_11_CSMS

Test case name	Update Charging Station Certificate by request of CSMS - Success - Charging Station Certificate	
Test case Id	TC_A_11_CSMS	
Use case Id(s)	A02 & F06	
Requirement(s)	A02.FR.11, A02.FR.14 & F06.FR.01	
System under test	CSMS	
Description	The CSMS is able to request the Charging Station to update its charging station certificate using the TriggerMessageRequest message.	
Purpose	To verify if the CSMS is able to request the Charging Station to update its Charging Station Certificate.	
Prerequisite(s)	The CSMS supports security profile 3	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	1. Execute Reusable State <i>RenewChargingStationCertificate</i>	
Tool validations	N/a	
	Post scenario validations: N/a	

Table 265. Test Case Id: TC_A_14_CSMS

Test case name	Update Charging Station Certificate by request of CSMS - Invalid certificate	
Test case Id	TC_A_14_CSMS	
Use case Id(s)	A02	
Requirement(s)	N/a	
System under test	CSMS	
Description	The CSMS is able to request the Charging Station to update its charging station certificate using the TriggerMessageRequest message.	
Purpose	To verify if the CSMS is able to handle a Charging Station rejecting the new Charging Station certificate.	
Prerequisite(s)	The CSMS supports security profile 3	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The OCTT responds with a TriggerMessageResponse With status Accepted	1. The CSMS sends a TriggerMessageRequest
	3 The OCTT sends a SignCertificateRequest With csr <Configured CSR> certificateType ChargingStationCertificate	4. The CSMS responds with a SignCertificateResponse
	6. The OCTT responds with a CertificateSignedResponse With status Rejected	5. The CSMS sends a CertificateSignedRequest
Tool validations	<ul style="list-style-type: none"> * Step 1: Message: TriggerMessageRequest - requestedMessage SignChargingStationCertificate * Step 4: Message: SignCertificateResponse - status Accepted <p>Post scenario validations: N/a</p>	

Table 266. Test Case Id: TC_A_19_CSMS

Test case name	Upgrade Charging Station Security Profile - Accepted	
Test case Id	TC_A_19_CSMS	
Use case Id(s)	A05	
Requirement(s)	A05.FR.04, A05.FR.07	
System under test	CSMS	
Description	The CSMS updates the connection details on the Charging Station, to increase the security profile level.	
Purpose	To verify if the CSMS is able to set a new network connection profile at one of the by the Charging Station defined configuration slots with a higher security profile than currently configured.	
Prerequisite(s)	<ul style="list-style-type: none"> - Security profile must be set to 1 or 2. - If Security profile is set to 1, then a trusted certificate must be installed. 	
Before (Preparations)	<p>Configuration State: N/a</p> <p>Memory State: If configured <Security profile> is 2, then <i>RenewChargingStationCertificate</i></p> <p>Reusable State(s): N/a</p>	
Main (Test scenario)	<p>Charging Station</p> <p><u>Manual Action:</u> Request the CSMS to set a new <i>NetworkConnectionProfile</i> with a security profile level one higher than currently configured</p> <p>2. The OCTT responds with a SetNetworkProfileResponse With status Accepted</p> <p><u>Manual Action:</u> Request the CSMS to change the <i>NetworkConfigurationPriority</i> to one that contains the <i>configurationSlot</i> of the new <i>NetworkConnectionProfile</i> from step 1</p> <p>4. The OCTT responds with a SetVariablesResponse with status Accepted</p> <p><u>Manual Action:</u> Request the CSMS to reboot the Charging Station</p> <p>6. The OCTT responds with a ResetResponse with status Accepted</p> <p>7. The OCTT reconnects to the CSMS with security profile is <Configured securityProfile + 1></p> <p>9. Execute Reusable State Booted</p> <p>10. The OCTT reconnects to the CSMS with security profile is <Configured securityProfile></p>	<p>CSMS</p> <p>1. The CSMS sends a SetNetworkProfileRequest</p> <p>3. The CSMS sends a SetVariablesRequest</p> <p>5. The CSMS sends a ResetRequest</p> <p>8. The CSMS accepts the connection attempt.</p> <p>11. The CSMS shall not accept the connection attempt.</p>
Tool validations	<ul style="list-style-type: none"> * Step 1: Message SetNetworkProfileRequest - connectionData.messageTimeout <Configured messageTimeout> - connectionData.ocppCsmsUrl <Configured ocppCsmsUrl> - connectionData.ocppInterface <Configured ocppInterface> - connectionData.ocppTransport JSON - connectionData.ocppVersion OCPP20 - connectionData.securityProfile <Configured securityProfile + 1> * Step 3: Message SetVariablesRequest setVariableData: - variable.name = "NetworkConfigurationPriority" - component.name = "OCPPCommCtrlr" - attributeValue = <contains configurationSlot provided at step 1> <p>Post scenario validations: - N/a</p>	

3.3. B Provisioning

Table 267. Test Case Id: TC_B_01_CSMS

Test case name	Cold Boot Charging Station - Accepted	
Test case Id	TC_B_01_CSMS	
Use case Id(s)	B01	
Requirement(s)	B01.FR.02	
System under test	CSMS	
Description	The booting mechanism allows a Charging Station to provide some general information about the Charging Station to the CSMS on startup AND it allows the Charging Station to request whether it is allowed to start sending other OCPP messages.	
Purpose	To verify whether the CSMS is able to accept the communications of a registered Charging Station.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	1. Execute Reusable State <i>Booted</i>	
Tool validations	N/a	
	Post scenario validations: N/a	

Table 268. Test Case Id: TC_B_02_CSMS

Test case name	Cold Boot Charging Station - Pending	
Test case Id	TC_B_02_CSMS	
Use case Id(s)	B02	
Requirement(s)	B02.FR.01, B02.FR.06	
System under test	CSMS	
Description	The booting mechanism allows a Charging Station to provide some general information about the Charging Station to the CSMS on startup AND it allows the Charging Station to request whether it is allowed to start sending other OCPP messages. The CSMS may respond to the BootNotificationRequest with status <i>Pending</i> . The <i>Pending</i> status can indicate that the CSMS wants to retrieve or set certain information on the Charging Station before it will accept the Charging Station.	
Purpose	To verify whether the CSMS is able to accept the communications of a registered Charging Station.	
Prerequisite(s)	The CSMS is configured to first respond to a BootNotificationRequest with status Pending .	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): N/a	
Main (Test scenario)	Charging Station 1. The OCTT sends a BootNotificationRequest with reason PowerUp chargingStation.model <Configured model> chargingStation.vendorName <Configured vendorName>	CSMS 2. The CSMS responds with a BootNotificationResponse
	<u>Note(s):</u> - If the interval in the BootNotificationResponse equals 0, the OCTT will wait <Configured heartbeatInterval> seconds, before sending another BootNotificationRequest. - If the interval in the BootNotificationResponse > 0, the OCTT will wait <Interval provided at the BootNotificationResponse> seconds, before sending another BootNotificationRequest. - During this interval, the CSMS may send messages to retrieve information from the Charging Station (as described in use cases B06, B07, B08) or change its configuration by SetVariablesRequest (as described in use case B05). The OCTT will respond to these messages.	
	3. The OCTT sends a BootNotificationRequest with reason PowerUp chargingStation.model <Configured model> chargingStation.vendorName <Configured vendorName>	4. The CSMS responds with a BootNotificationResponse
	5. The OCTT notifies the CSMS about the current state of all connectors. Message: StatusNotificationRequest with connectorStatus Available Message: NotifyEventRequest with trigger Delta actualValue "Available" component.name "Connector" variable.name "AvailabilityState"	6. The CSMS responds accordingly.

Test case name	Cold Boot Charging Station - Pending
Tool validations	<p>* Step 2: Message: BootNotificationResponse - status Pending</p> <p>* Step 3: Message: BootNotificationResponse - status Accepted</p> <p>Post scenario validations: N/a</p>

Table 269. Test Case Id: TC_B_30_CSMS

Test case name	Cold Boot Charging Station - Pending/Rejected - SecurityError	
Test case Id	TC_B_30_CSMS	
Use case Id(s)	B02/B03	
Requirement(s)	B02.FR.09, B03.FR.07	
System under test	CSMS	
Description	The booting mechanism allows a Charging Station to provide some general information about the Charging Station to the CSMS on startup AND it allows the Charging Station to request whether it is allowed to start sending other OCPP messages. The CSMS may respond to the BootNotificationRequest with status <i>Pending</i> or <i>Rejected</i> . During this state, the Charging Station is not allowed to send RPC Framework: CALL message that is NOT a BootNotificationRequest or in case of status <i>Pending</i> , a message triggered by one of the following messages: TriggerMessageRequest, GetBaseReportRequest, GetReportRequest.	
Purpose	To verify whether the CSMS is able to handle unauthorized messages from the Charging Station by responding with a SecurityError.	
Prerequisite(s)	The CSMS is configured to first respond to a BootNotificationRequest with status Pending or Rejected .	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	<p>1. The OCTT sends a BootNotificationRequest with reason PowerUp chargingStation.model <Configured model> chargingStation.vendorName <Configured vendorName></p> <p>3. The OCTT notifies the CSMS about the current state of all connectors.</p> <p>Message: StatusNotificationRequest with connectorStatus Available Message: NotifyEventRequest with trigger Delta actualValue "Available" component.name "Connector" variable.name "AvailabilityState"</p>	<p>2. The CSMS responds with a BootNotificationResponse</p> <p>4. The CSMS responds with RPC Framework: CALLERROR: SecurityError.</p>
Tool validations	<ul style="list-style-type: none"> * Step 2: Message: BootNotificationResponse - status Pending OR Rejected <p>Post scenario validations: N/a</p>	

Table 270. Test Case Id: TC_B_31_CSMS

Test case name	Cold Boot Charging Station - Pending/Rejected - TriggerMessage	
Test case Id	TC_B_31_CSMS	
Use case Id(s)	B02, F06	
Requirement(s)	N/a	
System under test	CSMS	
Description	The booting mechanism allows a Charging Station to provide some general information about the Charging Station to the CSMS on startup AND it allows the Charging Station to request whether it is allowed to start sending other OCPP messages.	
Purpose	To verify whether the CSMS is able to send a TriggerMessageRequest to trigger a BootNotificationRequest, before the interval expired.	
Prerequisite(s)	The CSMS is configured to first respond to a BootNotificationRequest with status Pending or Rejected .	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): N/a	
Main (Test scenario)	Charging Station 1. The OCTT sends a BootNotificationRequest with reason PowerUp chargingStation.model <Configured model> chargingStation.vendorName <Configured vendorName>	CSMS 2. The CSMS responds with a BootNotificationResponse
	4. The OCTT responds with a TriggerMessageResponse with status Accepted	3. The CSMS sends a TriggerMessageRequest
	5. The OCTT sends a BootNotificationRequest with reason Triggered chargingStation.model <Configured model> chargingStation.vendorName <Configured vendorName>	6. The CSMS responds with a BootNotificationResponse
	7. The OCTT notifies the CSMS about the current state of all connectors. Message: StatusNotificationRequest with connectorStatus Available Message: NotifyEventRequest with trigger Delta actualValue "Available" component.name "Connector" variable.name "AvailabilityState"	8. The CSMS responds accordingly.
Tool validations	* Step 2: Message: BootNotificationResponse - status Pending OR Rejected * Step 3: Message: TriggerMessageRequest - requestedMessage BootNotification * Step 6: Message: BootNotificationResponse - status Accepted Post scenario validations: N/a	

Table 271. Test Case Id: TC_B_06_CSMS

Test case name	Get Variables - single value	
Test case Id	TC_B_06_CSMS	
Use case Id(s)	B06	
Requirement(s)	B06.FR.01, B06.FR.02, B06.FR.03, B06.FR.04, B06.FR.10, B06.FR.11	
System under test	CSMS	
Description	Get the value of two of the required variables of OCPPCommCtrlr	
Purpose	To test getting single value using GetVariablesRequest for one of the mandatory component/variable combinations that must exist in the DM implementation.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	2. OCTT responds with: GetVariablesResponse	1. Manually request CSMS to get data for: - OCPPCommCtrlr.OfflineThreshold
Tool validations	<p>* Step 1:</p> <p>Message: GetVariablesRequest with (in arbitrary order)</p> <p>getVariableData[0]:</p> <ul style="list-style-type: none"> - attributeType is at least absent or attributeType = <i>Actual</i>, but <i>Target</i>, <i>MinSet</i>, and <i>MaxSet</i> are also allowed - variable.name = "OfflineThreshold" - component.name = "OCPPCommCtrlr" <p>Post scenario validations: Manually validate that CSMS has correctly read the requested variables.</p>	

Table 272. Test Case Id: TC_B_07_CSMS

Test case name	Get Variables - multiple values	
Test case Id	TC_B_07_CSMS	
Use case Id(s)	B06	
Requirement(s)	B06.FR.01, B06.FR.02, B06.FR.03	
System under test	CSMS	
Description	Get the value of two of the required variables of OCPPCommCtrlr	
Purpose	To test getting multiple values using GetVariablesRequest for one of the mandatory component/variable combinations that must exist in the DM implementation.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): N/a	
Main (Test scenario)	Charging Station 2. OCTT responds with: GetVariablesResponse	CSMS 1. Manually request CSMS to get data for: - OCPPCommCtrlr.OfflineThreshold - AuthCtrlr.AuthorizeRemoteStart
Tool validations	* Step 1: Message: GetVariablesRequest with (in arbitrary order) getVariableData[0]: - attributeType is at least absent or attributeType = <i>Actual</i> , but <i>Target</i> , <i>MinSet</i> , and <i>MaxSet</i> are also allowed - variable.name = "OfflineThreshold" - component.name = "OCPPCommCtrlr" getVariableData[1]: - attributeType is at least absent or attributeType = <i>Actual</i> , but <i>Target</i> , <i>MinSet</i> , and <i>MaxSet</i> are also allowed - variable.name = "AuthorizeRemoteStart" - component.name = "AuthCtrlr" Post scenario validations: Manually validate that CSMS has correctly read the requested variables.	

Table 273. Test Case Id: TC_B_09_CSMS

Test case name	Set Variables - single value	
Test case Id	TC_B_09_CSMS	
Use case Id(s)	B05	
Requirement(s)	B05.FR.01, B05.FR.02, B05.FR.03, B05.FR.10, B05.FR.12	
System under test	CSMS	
Description	Set the value of one of the required variables of OCPPCommCtrlr	
Purpose	To test setting a single value using SetVariablesRequest for one of the mandatory component/variable combinations that must exist in the DM implementation.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): N/a	
Main (Test scenario)	Charging Station 2. OCTT responds with: SetVariablesResponse	CSMS 1. Manually request CSMS to set data for: - OCPPCommCtrlr.OfflineThreshold
Tool validations	* Step 1: Message: SetVariablesRequest with (in arbitrary order): setVariableData[1]: - variable.name = "OfflineThreshold" - component.name = "OCPPCommCtrlr" - attributeValue = "123" - attributeType is absent or attributeType = <i>Actual</i> Post scenario validations: Manually validate that CSMS has correctly set the requested variables.	

Table 274. Test Case Id: TC_B_10_CSMS

Test case name	Set Variables - multiple values	
Test case Id	TC_B_10_CSMS	
Use case Id(s)	B05	
Requirement(s)	B05.FR.01, B05.FR.02, B05.FR.03	
System under test	CSMS	
Description	Set the value of two of the required variables of OCPPCommCtrlr	
Purpose	To test setting multiple values using SetVariablesRequest for one of the mandatory component/variable combinations that must exist in the DM implementation.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): N/a	
Main (Test scenario)	Charging Station 2. OCTT responds with: SetVariablesResponse	CSMS 1. Manually request CSMS to set data for: - OCPPCommCtrlr.OfflineThreshold - AuthCtrlr.AuthorizeRemoteStart+
Tool validations	* Step 1: Message: SetVariablesRequest with (in arbitrary order): setVariableData[1]: - variable.name = "OfflineThreshold" - component.name = "OCPPCommCtrlr" - attributeValue = "123" - attributeType is absent or attributeType = <i>Actual</i> setVariableData[2]: - variable.name = "AuthorizeRemoteStart" - component.name = "AuthCtrlr" - attributeValue = "false" - attributeType is absent or attributeType = <i>Actual</i> Post scenario validations: Manually validate that CSMS has correctly set the requested variables.	

Table 275. Test Case Id: TC_B_12_CSMS

Test case name	Get Base Report - ConfigurationInventory	
Test case Id	TC_B_12_CSMS	
Use case Id(s)	B07	
Requirement(s)	B07.FR.07	
System under test	CSMS	
Description	CSMS requests a ConfigurationInventory base report.	
Purpose	To test that CSMS supports the ConfigurationInventory base report.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	2. OCTT responds with: GetBaseReportResponse	1. Manually instruct CSMS to retrieve a ConfigurationInventory report.
Tool validations	<ul style="list-style-type: none"> * Step 1: <p>Message: GetBaseReportRequest with:</p> <ul style="list-style-type: none"> - requestId has integer value ≥ 0 - reportBase = <i>ConfigurationInventory</i> <p>Post scenario validations: CSMS receives all NotifyReportRequest message for this requestId and is able to show the result of configuration inventory to an operator.</p>	

Table 276. Test Case Id: TC_B_13_CSMS

Test case name	Get Base Report - FullInventory	
Test case Id	TC_B_13_CSMS	
Use case Id(s)	B07	
Requirement(s)	B07.FR.08	
System under test	CSMS	
Description	CSMS requests a FullInventory base report.	
Purpose	To test that CSMS supports the FullInventory base report.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	2. OCTT responds with: GetBaseReportResponse	1. Manually instruct CSMS to retrieve a FullInventory report.
Tool validations	<ul style="list-style-type: none"> * Step 1: GetBaseReportRequest with: <ul style="list-style-type: none"> - requestId has integer value ≥ 0 - reportBase = <i>FullInventory</i> <p>Post scenario validations: CSMS receives all NotifyReportRequest message for this requestId and is able to show the result of full inventory to an operator.</p>	

Table 277. Test Case Id: TC_B_20_CSMS

Test case name	Reset Charging Station - Without ongoing transaction - OnIdle	
Test case Id	TC_B_20_CSMS	
Use case Id(s)	B11	
Requirement(s)	B11.FR.04	
System under test	CSMS	
Description	This test case covers how the CSMS can request the Charging Station to reset itself by sending a ResetRequest without any ongoing transaction. This could for example be necessary if the Charging Station is not functioning correctly.	
Purpose	To verify if the CSMS is able to perform the reset mechanism as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): N/a	
Main (Test scenario)	Charging Station <u>Manual Action:</u> Request the CSMS to reboot the Charging Station with type _OnIdle 2. The OCTT responds with a ResetResponse with status Accepted 3. The OCTT sends a BootNotificationRequest 5. The OCTT notifies the CSMS about the current state of all connectors. Message: StatusNotificationRequest - connectorStatus Available Message: NotifyEventRequest - trigger Delta - actualValue "Available" - component.name "Connector" - variable.name "AvailabilityState"	CSMS 1. The CSMS sends a ResetRequest 4. The CSMS responds with a BootNotificationResponse 6. The CSMS responds accordingly.
Tool validations	* Step 4: Message BootNotificationResponse - status Accepted Post scenario validations: - N/a	

Table 278. Test Case Id: TC_B_21_CSMS

Test case name	Reset Charging Station - With Ongoing Transaction - OnIdle					
Test case Id	TC_B_21_CSMS					
Use case Id(s)	B12					
Requirement(s)	B12.FR.01, B12.FR.03, E07.FR.03					
System under test	CSMS					
Description	<p>This test case covers how the CSMS can remotely request the Charging Station to reset itself by sending a ResetRequest during a transaction. When ResetRequest "OnIdle" is send the charging stations schedules a reboot after all transactions are stopped.</p> <p>This could for example be necessary if the Charging Station is not functioning correctly.</p>					
Purpose	To verify if the CSMS is able to perform the reset mechanism as described at the OCPP specification.					
Prerequisite(s)	n/a					
Before (Preparations)	<p>Configuration State: N/a</p> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>EnergyTransferStarted</i></p>					
Main (Test scenario)	<table border="1"> <tr> <td>Charging Station</td> <td>CSMS</td> </tr> <tr> <td colspan="2"><i>Manual Action: Request the CSMS to reboot the Charging Station with status OnIdle</i></td></tr> </table> <p>2. The OCTT responds with a ResetResponse with status Scheduled</p> <p>3. The OCTT sends a TransactionEventRequest.</p> <ul style="list-style-type: none"> - eventType Updated - triggerReason StopAuthorized - transactionInfo.chargingState EVConnected - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type> <p>5. The OCTT sends a TransactionEventRequest.</p> <ul style="list-style-type: none"> - eventType Ended - triggerReason EVCommunicationLost - transactionInfo.chargingState Idle - transactionInfo.stoppedReason EVDisconnected <p>7. The OCTT sends a BootNotificationRequest with reason ScheduledReset</p> <p>9. The OCTT notifies the CSMS about the current state of all connectors.</p> <p>Message: StatusNotificationRequest</p> <ul style="list-style-type: none"> - connectorStatus Available <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none"> - trigger Delta - actualValue "Available" - component.name "Connector" - variable.name "AvailabilityState" 	Charging Station	CSMS	<i>Manual Action: Request the CSMS to reboot the Charging Station with status OnIdle</i>		<p>1. The CSMS sends a ResetRequest with status OnIdle</p> <p>4. The CSMS responds with a TransactionEventResponse.</p> <p>6. The CSMS responds with a TransactionEventResponse.</p> <p>8. The CSMS responds with a BootNotificationResponse</p> <p>10. The CSMS responds accordingly.</p>
Charging Station	CSMS					
<i>Manual Action: Request the CSMS to reboot the Charging Station with status OnIdle</i>						

Test case name	Reset Charging Station - With Ongoing Transaction - OnIdle
Tool validations	* Step 1: Message ResetRequest - type <i>OnIdle</i> * Step 8: Message BootNotificationResponse - status Accepted Post scenario validations: - N/a

Table 279. Test Case Id: TC_B_22_CSMS

Test case name	Reset Charging Station - With Ongoing Transaction - Immediate													
Test case Id	TC_B_22_CSMS													
Use case Id(s)	B12													
Requirement(s)	N/a													
System under test	CSMS													
Description	<p>This test case covers how the CSMS can remotely request the Charging Station to reset itself by sending a ResetRequest during a transaction. When ResetRequest "Immediate" is send the charging stations will try to stop all transactions before rebooting.</p> <p>This could for example be necessary if the Charging Station is not functioning correctly.</p>													
Purpose	To verify if the CSMS is able to perform the reset mechanism as described at the OCPP specification.													
Prerequisite(s)	n/a													
Before (Preparations)	<p>Configuration State: N/a</p> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>EnergyTransferStarted</i></p>													
Main (Test scenario)	<table border="1"> <tr> <td>Charging Station</td> <td>CSMS</td> </tr> <tr> <td colspan="2"><i>Manual Action: Request the CSMS to reboot the Charging Station with status Immediate</i></td> </tr> <tr> <td>2. The OCTT responds with a ResetResponse with status Accepted</td> <td>1. The CSMS sends a ResetRequest with status Immediate</td> </tr> <tr> <td>3. The OCTT sends a TransactionEventRequest. - eventType <i>Ended</i> - triggerReason <i>ResetCommand</i> - transactionInfo.chargingState <i>EVConnected</i> - transactionInfo.stoppedReason <i>ImmediateReset</i> - idToken is omitted</td> <td>4. The CSMS responds with a TransactionEventResponse.</td> </tr> <tr> <td>5. The OCTT sends a BootNotificationRequest with reason RemoteReset</td> <td>6. The CSMS responds with a BootNotificationResponse</td> </tr> <tr> <td>7. The OCTT notifies the CSMS about the current state of all connectors. For <Configured connectorId>: Message: StatusNotificationRequest - connectorStatus <i>Occupied</i> Message: NotifyEventRequest - trigger <i>Delta</i> - actualValue "<i>Occupied</i>" - component.name "<i>Connector</i>" - variable.name "<i>AvailabilityState</i>" For <Other connector(s)>: Message: StatusNotificationRequest - connectorStatus <i>Available</i> Message: NotifyEventRequest - trigger <i>Delta</i> - actualValue "<i>Available</i>" - component.name "<i>Connector</i>" - variable.name "<i>AvailabilityState</i>"</td> <td>8. The CSMS responds accordingly.</td> </tr> </table>	Charging Station	CSMS	<i>Manual Action: Request the CSMS to reboot the Charging Station with status Immediate</i>		2. The OCTT responds with a ResetResponse with status Accepted	1. The CSMS sends a ResetRequest with status Immediate	3. The OCTT sends a TransactionEventRequest . - eventType <i>Ended</i> - triggerReason <i>ResetCommand</i> - transactionInfo.chargingState <i>EVConnected</i> - transactionInfo.stoppedReason <i>ImmediateReset</i> - idToken is omitted	4. The CSMS responds with a TransactionEventResponse .	5. The OCTT sends a BootNotificationRequest with reason RemoteReset	6. The CSMS responds with a BootNotificationResponse	7. The OCTT notifies the CSMS about the current state of all connectors. For <Configured connectorId>: Message: StatusNotificationRequest - connectorStatus <i>Occupied</i> Message: NotifyEventRequest - trigger <i>Delta</i> - actualValue " <i>Occupied</i> " - component.name " <i>Connector</i> " - variable.name " <i>AvailabilityState</i> " For <Other connector(s)>: Message: StatusNotificationRequest - connectorStatus <i>Available</i> Message: NotifyEventRequest - trigger <i>Delta</i> - actualValue " <i>Available</i> " - component.name " <i>Connector</i> " - variable.name " <i>AvailabilityState</i> "	8. The CSMS responds accordingly.	
Charging Station	CSMS													
<i>Manual Action: Request the CSMS to reboot the Charging Station with status Immediate</i>														
2. The OCTT responds with a ResetResponse with status Accepted	1. The CSMS sends a ResetRequest with status Immediate													
3. The OCTT sends a TransactionEventRequest . - eventType <i>Ended</i> - triggerReason <i>ResetCommand</i> - transactionInfo.chargingState <i>EVConnected</i> - transactionInfo.stoppedReason <i>ImmediateReset</i> - idToken is omitted	4. The CSMS responds with a TransactionEventResponse .													
5. The OCTT sends a BootNotificationRequest with reason RemoteReset	6. The CSMS responds with a BootNotificationResponse													
7. The OCTT notifies the CSMS about the current state of all connectors. For <Configured connectorId>: Message: StatusNotificationRequest - connectorStatus <i>Occupied</i> Message: NotifyEventRequest - trigger <i>Delta</i> - actualValue " <i>Occupied</i> " - component.name " <i>Connector</i> " - variable.name " <i>AvailabilityState</i> " For <Other connector(s)>: Message: StatusNotificationRequest - connectorStatus <i>Available</i> Message: NotifyEventRequest - trigger <i>Delta</i> - actualValue " <i>Available</i> " - component.name " <i>Connector</i> " - variable.name " <i>AvailabilityState</i> "	8. The CSMS responds accordingly.													

Test case name	Reset Charging Station - With Ongoing Transaction - Immediate
Tool validations	<p>* Step 1: Message ResetRequest - type <i>Immediate</i></p> <p>* Step 6: Message BootNotificationResponse - status Accepted</p> <p>Post scenario validations: - N/a</p>

Table 280. Test Case Id: TC_B_25_CSMS

Test case name	Reset EVSE - Without ongoing transaction	
Test case Id	TC_B_25_CSMS	
Use case Id(s)	B11	
Requirement(s)	B11.FR.04	
System under test	CSMS	
Description	This test case covers how the CSMS can request the Charging Station to reset an EVSE by sending a ResetRequest without any ongoing transaction. This could for example be necessary if the Charging Station is not functioning correctly.	
Purpose	To verify if the CSMS is able to perform the reset mechanism as described at the OCPP specification.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): N/a	
Main (Test scenario)	Charging Station <u>Manual Action:</u> Request the CSMS to reboot an EVSE with status <i>OnIdle</i>	CSMS 1. The CSMS sends a ResetRequest with status <i>OnIdle</i> and evselID <Configured evselID> 2. The OCTT responds with a ResetResponse with status <i>Accepted</i>
Tool validations	* Step 1: Message ResetRequest - type <i>OnIdle</i> - evselID <Configured evselID> Post scenario validations: - N/a	

Table 281. Test Case Id: TC_B_26_CSMS

Test case name	Reset EVSE - With Ongoing Transaction - OnIdle					
Test case Id	TC_B_26_CSMS					
Use case Id(s)	B12					
Requirement(s)	B12.FR.07					
System under test	CSMS					
Description	<p>This test case covers how the CSMS can remotely request the Charging Station to reset an EVSE by sending a ResetRequest during a transaction. When ResetRequest "OnIdle" is send the charging stations schedules a reboot after all transactions are stopped.</p> <p>This could for example be necessary if the Charging Station is not functioning correctly.</p>					
Purpose	To verify if the CSMS is able to perform the reset mechanism as described at the OCPP specification.					
Prerequisite(s)	n/a					
Before (Preparations)	<p>Configuration State: N/a</p> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>EnergyTransferStarted</i></p>					
Main (Test scenario)	<table border="1"> <tr> <td>Charging Station</td> <td>CSMS</td> </tr> <tr> <td colspan="2">Manual Action: Request the CSMS to reboot the charging EVSE with status <i>OnIdle</i></td></tr> </table> <p>2. The OCTT responds with a ResetResponse with status Scheduled</p> <p>3. The OCTT sends a TransactionEventRequest.</p> <ul style="list-style-type: none"> - eventType Updated - triggerReason StopAuthorized - transactionInfo.chargingState EVConnected - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type> <p>5. The OCTT sends a TransactionEventRequest.</p> <ul style="list-style-type: none"> - eventType Ended - triggerReason EVCommunicationLost - transactionInfo.chargingState Idle - transactionInfo.stoppedReason EVDisconnected 	Charging Station	CSMS	Manual Action: Request the CSMS to reboot the charging EVSE with status <i>OnIdle</i>		<p>1. The CSMS sends a ResetRequest with status OnIdle and evseld <Configured evseld></p> <p>4. The CSMS responds with a TransactionEventResponse.</p> <p>6. The CSMS responds with a TransactionEventResponse.</p>
Charging Station	CSMS					
Manual Action: Request the CSMS to reboot the charging EVSE with status <i>OnIdle</i>						
Tool validations	<p>* Step 1: Message ResetRequest - type OnIdle - evseld <Configured evseld></p> <p>Post scenario validations: - N/a</p>					

Table 282. Test Case Id: TC_B_27_CSMS

Test case name	Reset EVSE - With Ongoing Transaction - Immediate	
Test case Id	TC_B_27_CSMS	
Use case Id(s)	B12	
Requirement(s)	N/a	
System under test	CSMS	
Description	<p>This test case covers how the CSMS can remotely request the Charging Station to reset an EVSE by sending a ResetRequest during a transaction. When ResetRequest "Immediate" is send the charging stations will try to stop all transactions before rebooting.</p> <p>This could for example be necessary if the Charging Station is not functioning correctly.</p>	
Purpose	To verify if the CSMS is able to perform the reset mechanism as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): State is <i>EnergyTransferStarted</i>	
Main (Test scenario)	Charging Station Manual Action: Request the CSMS to reboot the charging EVSE with status <i>Immediate</i>	CSMS
	2. The OCTT responds with a ResetResponse with status Accepted	1. The CSMS sends a ResetRequest with status Immediate and evseld <Configured evseld>
	3. The OCTT sends a TransactionEventRequest . - eventType <i>Ended</i> - triggerReason <i>ResetCommand</i> - transactionInfo.chargingState <i>EVConnected</i> - transactionInfo.stoppedReason <i>ImmediateReset</i>	4. The CSMS responds with a TransactionEventResponse.
Tool validations	* Step 1: Message ResetRequest - type <i>Immediate</i> - evseld <Configured evseld> Post scenario validations: N/a	

Table 283. Test Case Id: TC_B_42_CSMS

Test case name	Set new NetworkConnectionProfile - Accepted	
Test case Id	TC_B_42_CSMS	
Use case Id(s)	B09	
Requirement(s)	B09.FR.01	
System under test	CSMS	
Description	The CSMS updates the connection details on the Charging Station. For instance in preparation of a migration to a new CSMS.	
Purpose	To verify if the CSMS is able to set a new network connection profile at one of the by the Charging Station defined configuration slots.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The OCTT responds with a SetNetworkProfileResponse With status Accepted	1. The CSMS sends a SetNetworkProfileRequest
Tool validations	<ul style="list-style-type: none"> * Step 1: Message SetNetworkProfileRequest - configurationSlot is <i><Configured configurationSlot></i> - connectionData.messageTimeout <i><Configured messageTimeout></i> - connectionData.ocppCsmsUrl <i><Configured ocppCsmsUrl></i> - connectionData.ocppInterface <i><Configured ocppInterface></i> - connectionData.ocppTransport <i>JSON</i> - connectionData.ocppVersion <i>OCPP20</i> - connectionData.securityProfile <i><Configured securityProfile></i> 	
	Post scenario validations: <ul style="list-style-type: none"> - N/a 	

Table 284. Test Case Id: TC_B_44_CSMS

Test case name	Set new NetworkConnectionProfile - Failed	
Test case Id	TC_B_44_CSMS	
Use case Id(s)	B09	
Requirement(s)	B09.FR.03	
System under test	CSMS	
Description	The CSMS updates the connection details on the Charging Station. For instance in preparation of a migration to a new CSMS.	
Purpose	To verify if the CSMS is able to handle a Charging Station responding with status Failed, when setting a new network connection profile at one of the by the Charging Station defined configuration slots.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The OCTT responds with a SetNetworkProfileResponse With status Failed	1. The CSMS sends a SetNetworkProfileRequest
Tool validations	N/a	
	Post scenario validations: - N/a	

3.4. C Authorization

Table 285. Test Case Id: TC_C_02_CSMS

Test case name	Local start transaction - Authorization Invalid/Unknown	
Test case Id	TC_C_02_CSMS	
Use case Id(s)	C01, C04, C06	
Requirement(s)	C01.FR.07 OR C04.FR.01 OR C06.FR.04	
System under test	CSMS	
Description	When a Charging Station needs to charge an EV, it needs to authorize the EV Driver first at the CSMS before the charging can be started or stopped.	
Purpose	To verify whether the CSMS is able to report that an idToken is NOT valid.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	1. The OCTT sends an AuthorizeRequest with idToken.idToken <Configured invalid_idtoken_idtoken> idToken.type <Configured invalid_idtoken_type>	2. The CSMS responds with an AuthorizeResponse
Tool validations	<ul style="list-style-type: none"> * Step 2: Message: AuthorizeResponse - idTokenInfo.status <i>Invalid or Unknown</i> 	
	Post scenario validations: <ul style="list-style-type: none"> - N/a 	

Table 286. Test Case Id: TC_C_06_CSMS

Test case name	Local start transaction - Authorization Blocked	
Test case Id	TC_C_06_CSMS	
Use case Id(s)	C01	
Requirement(s)	C01.FR.07	
System under test	CSMS	
Description	When a Charging Station needs to charge an EV, it needs to authorize the EV Driver first at the CSMS before the charging can be started or stopped.	
Purpose	To verify whether the CSMS is able to report that an idToken is Blocked.	
Prerequisite(s)	N/a	
Before (Preparations)	<p>Configuration State: The IdToken configured as Blocked at the OCTT, must be set as Blocked at the CSMS.</p> <p>Memory State: N/a</p> <p>Reusable State(s): N/a</p>	
Main (Test scenario)	<p>Charging Station</p> <p>1. The OCTT sends an AuthorizeRequest with idToken.idToken <Configured blocked_idtoken_idtoken> idToken.type <Configured blocked_idtoken_type></p>	<p>CSMS</p> <p>2. The CSMS responds with an AuthorizeResponse</p>
Tool validations	<ul style="list-style-type: none"> * Step 2: Message: AuthorizeResponse - idTokenInfo.status Blocked or Invalid <p>Post scenario validations:</p>	

Table 287. Test Case Id: TC_C_07_CSMS

Test case name	Local start transaction - Authorization Expired	
Test case Id	TC_C_07_CSMS	
Use case Id(s)	C01	
Requirement(s)	C01.FR.07	
System under test	CSMS	
Description	When a Charging Station needs to charge an EV, it needs to authorize the EV Driver first at the CSMS before the charging can be started or stopped.	
Purpose	To verify whether the CSMS is able to report that an idToken is Expired.	
Prerequisite(s)	N/a	
Before (Preparations)	<p>Configuration State: The IdToken configured as Expired at the OCTT, must be set as Expired at the CSMS.</p> <p>Memory State: N/a</p> <p>Reusable State(s): N/a</p>	
Main (Test scenario)	<p>Charging Station</p> <p>1. The OCTT sends an AuthorizeRequest with idToken.idToken <Configured expired_idtoken_idtoken> idToken.type <Configured expired_idtoken_type></p>	<p>CSMS</p> <p>2. The CSMS responds with an AuthorizeResponse</p>
Tool validations	<ul style="list-style-type: none"> * Step 2: Message: AuthorizeResponse - idTokenInfo.status <i>Expired or Invalid</i> <p>Post scenario validations:</p>	

Table 288. Test Case Id: TC_C_08_CSMS

Test case name	Authorization through authorization cache - Accepted	
Test case Id	TC_C_08_CSMS	
Use case Id(s)	C12	
Requirement(s)	C12_FR_03	
System under test	CSMS	
Description	This test case describes how the EV Driver is authorized to start a transaction while the Charging Station uses Cached IdToken. This enables the EV Driver to Online start a transaction by using the Authorization Cache in which the Charging Station can respond faster, as no AuthorizeRequest is being sent.	
Purpose	To verify if the CSMS is able to respond correctly when an idToken which has status "Accepted" in the charging stations cache is presented according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	N/a	
Before (Preparations)	<p>Configuration State: N/a</p> <p>Memory State: N/a</p> <p>Charging State: State is <i>EVConnectedPreSession</i></p>	
Main (Test scenario)	<p>Charging Station</p> <p>1. The OCTT sends a TransactionEventRequest with</p> <ul style="list-style-type: none"> - triggerReason Authorized - idToken <Valid id token configured in Authorization Cache> - eventType Updated <p>Note(s):</p> <ul style="list-style-type: none"> - TxStartPoint contains <i>ParkingBayOccupancy</i> 	<p>CSMS</p> <p>2. The CSMS responds with a TransactionEventResponse</p>
Tool validations	<p>* Step 2: Message TransactionEventResponse - idTokenInfo.status Accepted</p> <p>Post scenario validations: - N/a</p>	

Table 289. Test Case Id: TC_C_20_CSMS

Test case name	Authorization through authorization cache - Invalid	
Test case Id	TC_C_20_CSMS	
Use case Id(s)	C12	
Requirement(s)	C12_FR_03	
System under test	CSMS	
Description	This test case describes how the EV Driver is authorized to start a transaction while the Charging Station uses Cached IdToken. This enables the EV Driver to Online start a transaction by using the Authorization Cache in which the Charging Station can respond faster, as no AuthorizeRequest is being sent.	
Purpose	To verify if the CSMS is able to respond correctly when an idToken, which has status "Invalid" in the charging stations cache but not in the CSMS, is presented according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Charging State: State is <i>EVConnectedPreSession</i>	
Main (Test scenario)	Charging Station 1. The OCTT sends a TransactionEventRequest with - triggerReason Authorized - idToken.idToken <Configured <i>invalid_idtoken_idtoken</i> > - idToken.type <Configured <i>invalid_idtoken_type</i> > - eventType Updated <u>Note(s):</u> - TxStartPoint contains <i>ParkingBayOccupancy</i>	CSMS 2. The CSMS responds with a TransactionEventResponse
Tool validations	* Step 2: Message TransactionEventResponse - idTokenInfo.status <i>Invalid</i> or <i>Unknown</i>	
	Post scenario validations: - N/a	

Table 290. Test Case Id: TC_C_37_CSMS

Test case name	Clear Authorization Data in Authorization Cache - Accepted	
Test case Id	TC_C_37_CSMS	
Use case Id(s)	C11	
Requirement(s)	N/a	
System under test	CSMS	
Description	This test case covers how the Charging Station autonomously stores a record of previously presented identifiers that have been successfully authorized by the CSMS in the Authorization Cache. (Successfully meaning: a response received on a message containing an IdToken)	
Purpose	To verify if the CSMS is able to request the Charging Station to clear all identifiers from the Authorization Cache according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	- N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The OCTT responds with a ClearCacheResponse with status Accepted	1. The CSMS sends a ClearCacheRequest
Tool validations	- N/a	
	Post scenario validations: - N/a	

Table 291. Test Case Id: TC_C_38_CSMS

Test case name	Clear Authorization Data in Authorization Cache - Rejected	
Test case Id	TC_C_38_CSMS	
Use case Id(s)	C11	
Requirement(s)	N/a	
System under test	CSMS	
Description	This test case covers how the Charging Station autonomously stores a record of previously presented identifiers that have been successfully authorized by the CSMS in the Authorization Cache. (Successfully meaning: a response received on a message containing an IdToken)	
Purpose	To verify if the CSMS is able to request the Charging Station to clear all identifiers from the Authorization Cache according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	- N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The OCTT responds with a ClearCacheResponse with status Rejected	1. The CSMS sends a ClearCacheRequest
Tool validations	- N/a	
	Post scenario validations: - N/a	

Table 292. Test Case Id: TC_C_39_CSMS

Test case name	Authorization by GroupId - Success	
Test case Id	TC_C_39_CSMS	
Use case Id(s)	C09	
Requirement(s)	C09_FR_02, C09_FR_03	
System under test	CSMS	
Description	This test case covers how a Charging Station can authorize an action for an EV Driver based on GroupId information. This could for example be used if 2 people regularly use the same EV: they can use their own IdToken (e.g. RFID card), and can deauthorize transactions that were started with the other idToken (with the same GroupId).	
Purpose	To verify if the CSMS is able to correctly handle the Authorization of idTokens with the same GroupId according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	- N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: Two valid idTokens with the same GroupId are configured	
	Reusable State(s): state is <i>EVConnectedPreSession</i>	
Main (Test scenario)	Charging Station 1. The OCTT sends an AuthorizeRequest with idToken.idToken <Configured valid_idtoken2_idtoken> idToken.type <Configured valid_idtoken2_type>	CSMS 2. The CSMS responds with an AuthorizeResponse
	3. The OCTT sends a TransactionEventRequest with - triggerReason Authorized - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type> if transaction was already started - eventType Updated else - eventType Started	4. The CSMS responds with a TransactionEventResponse
	5. Execute Reusable State <i>EnergyTransferStarted</i>	
	6. The OCTT sends an AuthorizeRequest with idToken.idToken <Configured valid_idtoken2_idtoken> idToken.type <Configured valid_idtoken2_type>	7. The CSMS responds with an AuthorizeResponse
	8. The OCTT sends a TransactionEventRequest with - triggerReason StopAuthorized - idToken.idToken <Configured valid_idtoken2_idtoken> - idToken.type <Configured valid_idtoken2_type> - eventType Updated	9. The CSMS responds with a TransactionEventResponse
	10. Execute Reusable State <i>EVConnectedPostSession</i>	
	11. Execute Reusable State <i>EVDDisconnected</i>	

Test case name	Authorization by GroupId - Success
Tool validations	<ul style="list-style-type: none"> * Step 2: Message AuthorizeResponse - idTokenInfo.status Accepted - idTokenInfo.groupIdToken.idToken <Configured groupIdToken> * Step 4: Message TransactionEventResponse - idTokenInfo.status Accepted - idTokenInfo.groupIdToken.idToken <Configured groupIdToken> * Step 7: Message AuthorizeResponse - idTokenInfo.status Accepted - idTokenInfo.groupIdToken.idToken <Configured groupIdToken> * Step 9: Message TransactionEventResponse - idTokenInfo.status Accepted - idTokenInfo.groupIdToken.idToken <Configured groupIdToken>
Post scenario validations:	<ul style="list-style-type: none"> - N/a

Table 293. Test Case Id: TC_C_47_CSMS

Test case name	Stop Transaction with a Master Pass - With UI - All transactions	
Test case Id	TC_C_47_CSMS	
Use case Id(s)	C16	
Requirement(s)	C16_FR_01	
System under test	CSMS	
Description	This test case covers how somebody with a Master Pass (User) can stop (selected) ongoing transactions, so the cable becomes unlocked. This Master Pass can be configured in: MasterPassGroupId. This could for example be usefull for Law Enforcement officials.	
Purpose	To verify if the CSMS is able to correctly respond on a request to stop all transactions when an idToken which has the MasterPass as GroupId is used and the user has selected to stop all transactions in the User Interface according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	- N/a	
Before (Preparations)	Configuration State: N/a Memory State: An idToken with the MastersPass as GroupId is configured Reusable State(s): State is <i>EnergyTransferStarted</i> for EVSE 1 with idToken valid idToken State is <i>EnergyTransferStarted</i> for EVSE 2 with idToken valid idToken2	
Main (Test scenario)	Charging Station 1. The OCTT sends an AuthorizeRequest with idToken.idToken <Configured masterpass_idtoken_idtoken> idToken.type <Configured masterpass_idtoken_type>	CSMS 2. The CSMS responds with an AuthorizeResponse
	3. The OCTT sends a TransactionEventRequest with - transactionInfo.stoppedReason MasterPass - idToken.idToken <Configured masterpass_idtoken_idtoken> - idToken.type <Configured masterpass_idtoken_type> - eventType Ended for both EVSE	4. The CSMS responds with a TransactionEventResponse for both EVSE

Table 294. Test Case Id: TC_C_48_CSMS

Test case name	Stop Transaction with a Master Pass - With UI - With UI - Specific transactions	
Test case Id	TC_C_48_CSMS	
Use case Id(s)	C16	
Requirement(s)	C16_FR_01	
System under test	CSMS	
Description	This test case covers how somebody with a Master Pass (User) can stop (selected) ongoing transactions, so the cable becomes unlocked. This Master Pass can be configured in: MasterPassGroupId. This could for example be usefull for Law Enforcement officials.	
Purpose	To verify if the CSMS is able to correctly respond on a request to stop a transaction when an idToken which has the MasterPass as GroupId is used and the user has selected to stop one transaction in the User Interface according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	- N/a	
Before (Preparations)	Configuration State: N/a Memory State: An idToken with the MastersPass as GroupId is configured Reusable State(s): State is <i>EnergyTransferStarted</i> for all EVSE	
Main (Test scenario)	Charging Station 1. The OCTT sends an AuthorizeRequest with idToken.idToken <Configured valid_idtoken_idtoken> idToken.type <Configured valid_idtoken_type>	CSMS 2. The CSMS responds with an AuthorizeResponse
	3. The OCTT sends a TransactionEventRequest with - transactionInfo.stoppedReason MasterPass - idToken.idToken <Configured masterpass_idtoken_idtoken> - idToken.type <Configured masterpass_idtoken_type> - eventType Ended	4. The CSMS responds with a TransactionEventResponse
Tool validations	* Step 2: Message AuthorizeResponse - idTokenInfo.status Accepted - idTokenInfo.groupIdToken.idToken <Configured masterPassGroupId> * Step 4: Message TransactionEventResponse - idTokenInfo.status Accepted - idTokenInfo.groupIdToken.idToken <Configured masterPassGroupId> Post scenario validations: - N/a	

Table 295. Test Case Id: TC_C_49_CSMS

Test case name	Stop Transaction with a Master Pass - Without UI	
Test case Id	TC_C_49_CSMS	
Use case Id(s)	C16	
Requirement(s)	C16_FR_02	
System under test	CSMS	
Description	This test case covers how somebody with a Master Pass (User) can stop (selected) ongoing transactions, so the cable becomes unlocked. This Master Pass can be configured in: MasterPassGroupId. This could for example be usefull for Law Enforcement officials.	
Purpose	To verify if the CSMS is able to correctly respond on a request to stop all transactions when an idToken which has the MasterPass as GroupId is used and the Charging Station does not have a User Interface according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	- N/a	
Before (Preparations)	<p>Configuration State: N/a</p> <p>Memory State: An idToken with the MastersPass as GroupId is configured</p> <p>Reusable State(s): State is <i>EnergyTransferStarted</i> for EVSE 1 with idToken valid idToken State is <i>EnergyTransferStarted</i> for EVSE 2 with idToken valid idToken2</p>	
Main (Test scenario)	<p>Charging Station</p> <p>1. The OCTT sends an AuthorizeRequest with idToken.idToken <Configured masterpass_idtoken_idtoken> idToken.type <Configured masterpass_idtoken_type></p> <p>3. The OCTT sends a TransactionEventRequest with - transactionInfo.stoppedReason MasterPass - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type> - eventType Ended for both EVSE</p>	<p>CSMS</p> <p>2. The CSMS responds with an AuthorizeResponse</p> <p>4. The CSMS responds with a TransactionEventResponse for both EVSE</p>

3.5. D Local Authorization List Management

This section is intentionally blank, this will be added in a later version.

3.6. E Transactions

Table 296. Test Case Id: TC_E_01_CSMS

Test case name	Start transaction options - PowerPathClosed	
Test case Id	TC_E_01_CSMS	
Use case Id(s)	E01(S5)	
Requirement(s)	E01.FR.05	
System under test	CSMS	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the start options.	
Purpose	To verify if the CSMS is able to handle a Charging Station that starts a transaction when the power path has been closed.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	1. The OCTT sends an AuthorizeRequest With idToken.idToken <Configured valid_idtoken_idtoken> idToken.type <Configured valid_idtoken_type>	2. The CSMS responds with an AuthorizeResponse
	3. The OCTT notifies the CSMS about the status change of the connector. Message: StatusNotificationRequest - connectorStatus is <i>Occupied</i> Message: NotifyEventRequest - trigger is <i>Delta</i> - actualValue is <i>Occupied</i> - component.name is <i>Connector</i> - variable.name is <i>AvailabilityState</i>	4. The CSMS responds accordingly.
	5. The OCTT sends a TransactionEventRequest With eventType is <i>Started</i> triggerReason is <i>ChargingStateChanged</i> idToken.idToken <Configured valid_idtoken_idtoken> idToken.type <Configured valid_idtoken_type> evse.id is <Configured evselid> evse.connectorId is <Configured connectorId> transactionInfo.chargingState is <i>SuspendedEVSE</i>	6. The CSMS responds with a TransactionEventResponse
	7. The OCTT sends a TransactionEventRequest With eventType is <i>Updated</i> triggerReason is <i>ChargingStateChanged</i> transactionInfo.chargingState is <i>Charging</i>	8. The CSMS responds with a TransactionEventResponse

Test case name	Start transaction options - PowerPathClosed
Tool validations	<p>* Step 2: Message: AuthorizeResponse - idTokenInfo.status must be Accepted</p> <p>* Step 6: Message: TransactionEventResponse - idTokenInfo.status must be Accepted</p> <p>Post scenario validations: N/a</p>

Table 297. Test Case Id: TC_E_02_CSMS

Test case name	Start transaction options - EnergyTransfer	
Test case Id	TC_E_02_CSMS	
Use case Id(s)	E01(S6)	
Requirement(s)	E01.FR.06	
System under test	CSMS	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the start options.	
Purpose	To verify if the CSMS is able to handle a Charging Station that starts a transaction when the energy transfer starts.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): N/a	
Main (Test scenario)	Charging Station 1. The OCTT sends an AuthorizeRequest With idToken.idToken <Configured valid_idtoken_idtoken> idToken.type <Configured valid_idtoken_type> 3. The OCTT notifies the CSMS about the status change of the connector. Message: StatusNotificationRequest - connectorStatus is Occupied Message: NotifyEventRequest - trigger is Delta - actualValue is Occupied - component.name is Connector - variable.name is AvailabilityState	CSMS 2. The CSMS responds with an AuthorizeResponse 4. The CSMS responds accordingly.
	 5. The OCTT sends a TransactionEventRequest With eventType is Started triggerReason is ChargingStateChanged idToken.idToken <Configured valid_idtoken_idtoken> idToken.type <Configured valid_idtoken_type> evse.id is <Configured evseld> evse.connectorId is <Configured connectorId> transactionInfo.chargingState is Charging	6. The CSMS responds with a TransactionEventResponse
Tool validations	<ul style="list-style-type: none"> * Step 2: Message: AuthorizeResponse - idTokenInfo.status must be Accepted * Step 6: Message: TransactionEventResponse - idTokenInfo.status must be Accepted <p>Post scenario validations: N/a</p>	

Table 298. Test Case Id: TC_E_03_CSMS

Test case name	Local start transaction - Cable plugin first - Success	
Test case Id	TC_E_03_CSMS	
Use case Id(s)	E02	
Requirement(s)	E02.FR.02	
System under test	CSMS	
Description	OCPP 2.x.x allows an EV driver to either first connect the EV and EVSE OR present a form of identification. Both sequences will result in being able to charge.	
Purpose	To verify if the CSMS is able to handle a Charging Station that is able to start a charging session when the EV driver first connects the EV and EVSE, before authorization.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): State is <i>EVConnectedPreSession</i>	
Main (Test scenario)	Charging Station	CSMS
	1. Execute Reusable State <i>Authorized</i> 2. Execute Reusable State <i>EnergyTransferStarted</i>	
Tool validations	N/a	
	Post scenario validations: N/a	

Table 299. Test Case Id: TC_E_04_CSMS

Test case name	Local start transaction - Authorization first - Success	
Test case Id	TC_E_04_CSMS	
Use case Id(s)	E03	
Requirement(s)	E03.FR.02	
System under test	CSMS	
Description	OCPP 2.x.x allows an EV driver to either first connect the EV and EVSE OR present a form of identification. Both sequences will result in being able to charge.	
Purpose	To verify if the CSMS is able to handle a Charging Station that is able to start a charging session when the EV driver first presents a form of identification, before connecting the EV and EVSE.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	1. Execute Reusable State <i>Authorized</i>	
	2. Execute Reusable State <i>EnergyTransferStarted</i>	
Tool validations	N/a	
	Post scenario validations: N/a	

Table 300. Test Case Id: TC_E_39_CSMS

Test case name	Stop transaction options - Deauthorized - timeout	
Test case Id	TC_E_39_CSMS	
Use case Id(s)	E03, E06	
Requirement(s)	E03.FR.05, E06.FR.04	
System under test	CSMS	
Description	OCPP 2.x.x allows an EV driver to either first connect the EV and EVSE OR present a form of identification. Both sequences will result in being able to charge.	
Purpose	To verify if the CSMS is able to handle a Charging Station that deauthorizes the transaction after the EVConnectionTimeout has expired.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): State is <i>Authorized</i>	
Main (Test scenario)	Charging Station 1. The OCTT sends a TransactionEventRequest With triggerReason is <i>EVConnectTimeout</i> transactionInfo.stoppedReason is <i>Timeout</i> eventType is <i>Ended</i> <u>Note(s):</u> - This step will be executed after the _<Configured EV connection timeout> expires._ CSMS 2. The CSMS responds with a TransactionEventResponse	
Tool validations	N/a Post scenario validations: N/a	

Table 301. Test Case Id: TC_E_14_CSMS

Test case name	Stop transaction options - EVDisconnected - Charging Station side	
Test case Id	TC_E_14_CSMS	
Use case Id(s)	E06(S2)	
Requirement(s)	E06.FR.02	
System under test	CSMS	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the stop options.	
Purpose	To verify if the CSMS is able to handle a Charging Station that stops a transaction when the EV and EVSE are disconnected at the Charging Station side.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): State is <i>EVConnectedPostSession</i>	
Main (Scenario)	Charging Station	CSMS
	1. Execute Reusable State <i>EVDisconnected</i>	
Tool validations	N/a	
	Post scenario validations: N/a	

Table 302. Test Case Id: TC_E_20_CSMS

Test case name	Stop transaction options - EVDisconnected - EV side (able to charge IEC 61851-1 EV)	
Test case Id	TC_E_20_CSMS	
Use case Id(s)	E06(S2), E10	
Requirement(s)	E06.FR.02	
System under test	CSMS	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the stop options.	
Purpose	To verify if the CSMS is able to handle a Charging Station that stops a transaction when the EV and EVSE are disconnected at the EV side.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): State is <i>EnergyTransferSuspended</i>	
Main (Scenario)	Charging Station	CSMS
	1. Execute Reusable State <i>EVDisconnected</i>	
Tool validations	N/a	
	Post scenario validations: N/a	

Table 303. Test Case Id: TC_E_15_CSMS

Test case name	Stop transaction options - StopAuthorized - Local	
Test case Id	TC_E_15_CSMS	
Use case Id(s)	E06(S3)	
Requirement(s)	E06.FR.03	
System under test	CSMS	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the stop options.	
Purpose	To verify if the CSMS is able to handle a Charging Station that stops a transaction when the EV driver locally stops the transaction.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): State is <i>EnergyTransferStarted</i>	
Main (Test scenario)	Charging Station 1. The OCTT sends a TransactionEventRequest With triggerReason is <i>StopAuthorized</i> transactionInfo.stoppedReason is <i>Local</i> eventType is <i>Ended</i>	CSMS 2. The CSMS responds with a TransactionEventResponse
Tool validations	N/a	
	Post scenario validations: N/a	

Table 304. Test Case Id: TC_E_21_CSMS

Test case name	Stop transaction options - StopAuthorized - Remote	
Test case Id	TC_E_21_CSMS	
Use case Id(s)	E06(S3) AND F03	
Requirement(s)	E06.FR.03,F03.FR.01,F03.FR.09	
System under test	CSMS	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the stop options.	
Purpose	To verify if the CSMS is able to handle a Charging Station that stops a transaction when it receives a RequestStopTransactionRequest.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): State is <i>EnergyTransferStarted</i>	
Main (Test scenario)	Charging Station <u>Manual Action:</u> Trigger the CSMS to request the Charging Station to stop the ongoing transaction.	CSMS 1. The CSMS sends a RequestStopTransactionRequest
	2. The OCTT responds with a RequestStopTransactionResponse with status Accepted	4. The CSMS responds with a TransactionEventResponse .
	3. The OCTT sends a TransactionEventRequest . with triggerReason is <i>RemoteStop</i> transactionInfo.stoppedReason is <i>Remote</i> eventType is <i>Ended</i>	
Tool validations	* Step 1: Message: RequestStopTransactionRequest - transactionId must equal < transactionId provided by the OCTT in before state.> Post scenario validations: N/a	

Table 305. Test Case Id: TC_E_09_CSMS

Test case name	Start transaction options - EVConnected	
Test case Id	TC_E_09_CSMS	
Use case Id(s)	E01(S2)	
Requirement(s)	E01.FR.02	
System under test	CSMS	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the start options.	
Purpose	To verify if the CSMS is able to handle a Charging Station that starts a transaction when the EV and EVSE are connected.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): N/a	
Main (Test scenario)	Charging Station 1. The OCTT notifies the CSMS about the status change of the connector. Message: StatusNotificationRequest - connectorStatus is <i>Occupied</i> Message: NotifyEventRequest - trigger is <i>Delta</i> - actualValue is <i>Occupied</i> - component.name is <i>Connector</i> - variable.name is <i>AvailabilityState</i>	CSMS 2. The CSMS responds accordingly.
	 3. The OCTT sends a TransactionEventRequest With eventType is <i>Started</i> triggerReason is <i>CablePluggedIn</i> evse.id is < <i>Configured evseld</i> > evse.connectorId is < <i>Configured connectorId</i> > transactionInfo.chargingState is <i>EVConnected</i>	 4. The CSMS responds with a TransactionEventResponse
Tool validations	N/a Post scenario validations: N/a	

Table 306. Test Case Id: TC_E_10_CSMS

Test case name	Start transaction options - Authorized - Local	
Test case Id	TC_E_10_CSMS	
Use case Id(s)	E01(S3)	
Requirement(s)	E01.FR.03	
System under test	CSMS	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the start options.	
Purpose	To verify if the CSMS is able to handle a Charging Station that starts a transaction when the EV and EVSE are connected.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): N/a	
Main (Test scenario)	Charging Station 1. The OCTT sends an AuthorizeRequest With idToken.idToken <Configured valid_idtoken_idtoken> idToken.type <Configured valid_idtoken_type> 3. The OCTT sends a TransactionEventRequest With eventType is Started triggerReason is Authorized idToken.idToken <Configured valid_idtoken_idtoken> idToken.type <Configured valid_idtoken_type>	CSMS 2. The CSMS responds with an AuthorizeResponse 4. The CSMS responds with a TransactionEventResponse
Tool validations	<ul style="list-style-type: none"> * Step 2: Message: AuthorizeResponse - idTokenInfo.status must be Accepted * Step 4: Message: TransactionEventResponse - idTokenInfo.status must be Accepted <p>Post scenario validations: N/a</p>	

Table 307. Test Case Id: TC_E_11_CSMS

Test case name	Start transaction options - DataSigned	
Test case Id	TC_E_11_CSMS	
Use case Id(s)	E01(S4)	
Requirement(s)	E01.FR.04	
System under test	CSMS	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the start options.	
Purpose	To verify if the CSMS is able to handle a Charging Station that starts a transaction when the signed meter values are received.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): N/a	
Main (Test scenario)	Charging Station 1. The OCTT sends an AuthorizeRequest With idToken.idToken <Configured valid_idtoken_idtoken> idToken.type <Configured valid_idtoken_type> 3. The OCTT notifies the CSMS about the status change of the connector. Message: StatusNotificationRequest - connectorStatus is Occupied Message: NotifyEventRequest - trigger is Delta - actualValue is Occupied - component.name is Connector - variable.name is AvailabilityState	CSMS 2. The CSMS responds with an AuthorizeResponse 4. The CSMS responds accordingly.
	5. The OCTT sends a TransactionEventRequest With eventType is Started triggerReason is SignedDataReceived idToken.idToken <Configured valid_idtoken_idtoken> idToken.type <Configured valid_idtoken_type> evse.id is <Configured evseld> evse.connectorId is <Configured connectorId> meterValue is provided with the following values: sampledValue.value is 0.0 sampledValue.context is Transaction.Begin sampledValue.signedMeterValue is <Generated SignedMeterValueType>	6. The CSMS responds with a TransactionEventResponse
	7. The OCTT sends a TransactionEventRequest With eventType is Updated triggerReason is ChargingStateChanged transactionInfo.chargingState is Charging	8. The CSMS responds with a TransactionEventResponse

Test case name	Start transaction options - DataSigned
Tool validations	<p>* Step 2: Message: AuthorizeResponse - idTokenInfo.status must be Accepted</p> <p>* Step 6: Message: TransactionEventResponse - idTokenInfo.status must be Accepted</p> <p>Post scenario validations: N/a</p>

Table 308. Test Case Id: TC_E_12_CSMS

Test case name	Start transaction options - ParkingBayOccupied	
Test case Id	TC_E_12_CSMS	
Use case Id(s)	E01(S1)	
Requirement(s)	E01.FR.01	
System under test	CSMS	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the start options.	
Purpose	To verify if the CSMS is able to handle a Charging Station that starts a transaction when the EV and EVSE are connected.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Scenario)	Charging Station	CSMS
	1. The OCTT sends a TransactionEventRequest With eventType is Started triggerReason is EVDetected	2. The CSMS responds with a TransactionEventResponse
Tool validations	N/a	
	Post scenario validations: N/a	

Table 309. Test Case Id: TC_E_38_CSMS

Test case name	Local start transaction - EV not ready	
Test case Id	TC_E_38_CSMS	
Use case Id(s)	E03	
Requirement(s)	N/a	
System under test	CSMS	
Description	OCPP 2.x.x allows an EV driver to either first connect the EV and EVSE OR present a form of identification. Both sequences will result in being able to charge.	
Purpose	To verify if the CSMS is able to handle a Charging Station that reports an EV is not ready to start the energy transfer (yet).	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): State is <i>Authorized</i>	
Main (Test scenario)	Charging Station 1. Execute Reusable State <i>EVConnectedPreSession</i> 2. The OCTT sends a TransactionEventRequest With triggerReason is <i>ChargingStateChanged</i> transactionInfo.chargingState is <i>SuspendedEV</i> eventType is <i>Updated</i>	CSMS 3. The CSMS responds with a TransactionEventResponse
Tool validations	N/a Post scenario validations: N/a	

Table 310. Test Case Id: TC_E_07_CSMS

Test case name	Stop transaction options - PowerPathClosed - Local stop	
Test case Id	TC_E_07_CSMS	
Use case Id(s)	E06(S5)	
Requirement(s)	E06.FR.06	
System under test	CSMS	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the stop options.	
Purpose	To verify if the CSMS is able to handle a Charging Station that stops a transaction when it is locally stopped by an EV driver.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): State is <i>EnergyTransferStarted</i>	
Main (Test scenario)	Charging Station 1. The OCTT sends a TransactionEventRequest With triggerReason is <i>StopAuthorized</i> transactionInfo.stoppedReason is <i>Local</i> eventType is <i>Ended</i>	CSMS 2. The CSMS responds with a TransactionEventResponse
Tool validations	N/a	
	Post scenario validations: N/a	

Table 311. Test Case Id: TC_E_08_CSMS

Test case name	Stop transaction options - EnergyTransfer stopped - StopAuthorized	
Test case Id	TC_E_08_CSMS	
Use case Id(s)	E06(S6)	
Requirement(s)	E06.FR.07	
System under test	CSMS	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the stop options.	
Purpose	To verify if the CSMS is able to handle a Charging Station that stops a transaction when the energy transfer stopped normally.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): State is <i>StopAuthorized</i>	
Main (Test scenario)	Charging Station 1. The OCTT sends a TransactionEventRequest With triggerReason is <i>ChargingStateChanged</i> transactionInfo.chargingState is <i>EVConnected</i> transactionInfo.stoppedReason is <i>Local</i> eventType is <i>Ended</i>	CSMS 2. The CSMS responds with a TransactionEventResponse
Tool validations	N/a	
	Post scenario validations: N/a	

Table 312. Test Case Id: TC_E_16_CSMS

Test case name	Stop transaction options - Deauthorized - Invalid idToken	
Test case Id	TC_E_16_CSMS	
Use case Id(s)	E06(S3)	
Requirement(s)	E06.FR.04,E01.FR.11,E01.FR.12	
System under test	CSMS	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the start options.	
Purpose	To verify if the CSMS is able to handle a Charging Station that stops a transaction when the transaction gets deauthorized by the status from the idTokenInfo at a TransactionEventResponse message and it has been configured to do so.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): N/a	
Main (Scenario)	Charging Station 1. The OCTT sends a TransactionEventRequest With triggerReason is <i>Authorized</i> idToken.idToken < <i>Configured invalid_idtoken_idtoken</i> > idToken.type < <i>Configured invalid_idtoken_type</i> > eventType is <i>Started</i>	CSMS 2. The CSMS responds with a TransactionEventResponse
	 3. The OCTT sends a TransactionEventRequest With eventType <i>Ended</i> triggerReason <i>Deauthorized</i> transactionInfo.stoppedReason <i>DeAuthorized</i>	 4. The CSMS responds with a TransactionEventResponse
Tool validations	* Step 2: Message: TransactionEventResponse - idTokenInfo.status must be <i>Invalid</i> or <i>Unknown</i> + Post scenario validations: N/a	

Table 313. Test Case Id: TC_E_17_CSMS

Test case name	Stop transaction options - Deauthorized - EV side disconnect	
Test case Id	TC_E_17_CSMS	
Use case Id(s)	E06(S3)	
Requirement(s)	E06.FR.04	
System under test	CSMS	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the start options.	
Purpose	To verify if the CSMS is able to handle a Charging Station that stops a transaction when the transaction gets deauthorized by a connection loss from the EV side and it has been configured to do so.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): State is <i>EnergyTransferSuspended</i>	
Main (Scenario)	Charging Station 1. The OCTT sends a TransactionEventRequest triggerReason must be <i>EVCommunicationLost</i> transactionInfo.chargingState must be <i>Idle</i> transactionInfo.stoppedReason must be <i>EVDisconnected</i> eventType must be <i>Ended</i>	CSMS 2. The CSMS responds with a TransactionEventResponse
Tool validations	N/a Post scenario validations: N/a	

Table 314. Test Case Id: TC_E_22_CSMS

Test case name	Stop transaction options - EnergyTransfer stopped - SuspendedEV	
Test case Id	TC_E_22_CSMS	
Use case Id(s)	E06(S6)	
Requirement(s)	E06.FR.07	
System under test	CSMS	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the stop options.	
Purpose	To verify if the CSMS is able to handle a Charging Station that stops a transaction when the energy transfer stopped by the EV.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): State is <i>EnergyTransferStarted</i>	
Main (Test scenario)	Charging Station 1. The OCTT sends a TransactionEventRequest With triggerReason is <i>ChargingStateChanged</i> transactionInfo.chargingState is <i>SuspendedEV</i> transactionInfo.stoppedReason is <i>StoppedByEV</i> eventType is <i>Ended</i>	CSMS 2. The CSMS responds with a TransactionEventResponse
Tool validations	N/a	
	Post scenario validations: N/a	

Table 315. Test Case Id: TC_E_19_CSMS

Test case name	Stop transaction options - ParkingBayUnoccupied	
Test case Id	TC_E_19_CSMS	
Use case Id(s)	E06(S1)	
Requirement(s)	E06.FR.01	
System under test	CSMS	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the stop options.	
Purpose	To verify if the CSMS is able to handle a Charging Station that stops a transaction when the EV left the parking bay.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): State is <i>EVDisconnected</i>	
Main (Test scenario)	Charging Station 1. The OCTT sends a TransactionEventRequest With triggerReason is <i>EVDeparted</i> transactionInfo.stoppedReason is <i>Local</i> eventType is <i>Ended</i>	CSMS 2. The CSMS responds with a TransactionEventResponse
Tool validations	N/a	
	Post scenario validations: N/a	

Table 316. Test Case Id: TC_E_26_CSMS

Test case name	Disconnect cable on EV-side - Suspend transaction	
Test case Id	TC_E_26_CSMS	
Use case Id(s)	E10	
Requirement(s)	E10.FR.01	
System under test	CSMS	
Description	The Charging Station can behave in several different ways when the cable is disconnected at the EV side, based on its configuration. This test case tests one of the possible configuration settings.	
Purpose	To verify if the CSMS can handle a Charging Station that suspends the transaction when the EV and EVSE are disconnected at the EV side AND is able restart the energy transfer after reconnecting the EV and EVSE.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): State is <i>EnergyTransferSuspended</i>	
Main (Scenario)	Charging Station 1. The OCTT sends a TransactionEventRequest With triggerReason is <i>EVCommunicationLost</i> transactionInfo.chargingState is <i>Idle</i> eventType is <i>Updated</i> 3. The OCTT notifies the CSMS about the current state of the connector. Message: StatusNotificationRequest - connectorStatus <i>Available</i> - evseld <Configured evseld> - connectorId <Configured connectorId> Message: NotifyEventRequest - trigger <i>Delta</i> - actualValue "Available" - component.name "Connector" - component.evse.id <Configured evseld> - component.evse.connectorid <Configured connectorId> - variable.name "AvailabilityState"	CSMS 2. The CSMS responds with a TransactionEventResponse 4. The CSMS responds accordingly.
	5. The OCTT sends a TransactionEventRequest With triggerReason is <i>CablePluggedIn</i> transactionInfo.chargingState is <i>EVConnected</i> eventType is <i>Updated</i>	6. The CSMS responds with a TransactionEventResponse
	7. The OCTT sends a TransactionEventRequest With triggerReason is <i>ChargingStateChanged</i> transactionInfo.chargingState is <i>Charging</i> eventType is <i>Updated</i>	8. The CSMS responds with a TransactionEventResponse
Tool validations	N/a Post scenario validations: N/a	

Table 317. Test Case Id: TC_E_29_CSMS

Test case name	Check Transaction status - Transaction with id ongoing - with message in queue	
Test case Id	TC_E_29_CSMS	
Use case Id(s)	E14	
Requirement(s)	E14.FR.02,E14.FR.04	
System under test	CSMS	
Description	The CSMS is able to request the status of a transaction and to find out whether there are queued transaction-related messages, using the GetTransactionStatusRequest message.	
Purpose	To verify if the CSMS is able to request the status of queued TransactionEventRequest messages from a specific transaction by sending a GetTransactionStatusRequest with a transactionId. The OCTT will respond that there are message(s) queued belonging to the ongoing transaction with the requested id.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): State is <i>EnergyTransferStarted</i>	
Main (Test scenario)	Charging Station 1. The OCTT closes the WebSocket connection. 2. The OCTT waits a number of seconds equal to _<Configured Transaction Duration>, then it will reconnect to the CSMS._ 4. The OCTT responds with a GetTransactionStatusResponse With ongoingIndicator is true messagesInQueue is true 5. The OCTT sends a TransactionEventRequest With eventType is <i>Updated</i> meterValues is present. offline is true	CSMS 3. The CSMS sends a GetTransactionStatusRequest 6. The CSMS responds with a TransactionEventResponse
Tool validations	* Step 3: Message: GetTransactionStatusRequest - transactionId <Generated transactionId from Before> Post scenario validations: N/a	

Table 318. Test Case Id: TC_E_30_CSMS

Test case name	Check Transaction status - Transaction with id ongoing - without message in queue	
Test case Id	TC_E_30_CSMS	
Use case Id(s)	E14	
Requirement(s)	E14.FR.02,E14.FR.05	
System under test	CSMS	
Description	The CSMS is able to request the status of a transaction and to find out whether there are queued transaction-related messages, using the GetTransactionStatusRequest message.	
Purpose	To verify if the CSMS is able to request the status of queued TransactionEventRequest messages from a specific transaction by sending a GetTransactionStatusRequest with a transactionId. The OCTT will respond that there is NO message queued belonging to the ongoing transaction with the requested id.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): State is <i>EnergyTransferStarted</i>	
Main (Test scenario)	Charging Station	CSMS
	2. The OCTT responds with a GetTransactionStatusResponse With ongoingIndicator is <i>true</i> messagesInQueue is <i>false</i>	1. The CSMS sends a GetTransactionStatusRequest
Tool validations	<ul style="list-style-type: none"> * Step 1: Message: GetTransactionStatusRequest - transactionId must be <Generated transactionId from Before> 	
	Post scenario validations: N/a	

Table 319. Test Case Id: TC_E_31_CSMS

Test case name	Check Transaction status - Transaction with id ended - with message in queue	
Test case Id	TC_E_31_CSMS	
Use case Id(s)	E14	
Requirement(s)	E14.FR.03,E14.FR.04	
System under test	CSMS	
Description	The CSMS is able to request the status of a transaction and to find out whether there are queued transaction-related messages, using the GetTransactionStatusRequest message.	
Purpose	To verify if the CSMS is able to request the status of queued TransactionEventRequest messages from a specific transaction by sending a GetTransactionStatusRequest with a transactionId. The OCTT will respond that there are message(s) queued belonging to an ended transaction with the requested id.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): State is <i>EnergyTransferStarted</i>	
Main (Test scenario)	Charging Station 1. The OCTT closes the WebSocket connection. 2. The OCTT waits a number of seconds equal to _<Configured Transaction duration>, then it will reconnect to the CSMS._ 3. The OCTT sends a TransactionEventRequest With eventType is <i>Ended</i> offline is <i>true</i> triggerReason is <i>EVCommunicationLost</i> transactionInfo.chargingState is <i>Idle</i> seqNo <Skips two sequence number values>	CSMS 4. The CSMS responds with a TransactionEventResponse
	6. The OCTT responds with a GetTransactionStatusResponse With ongoingIndicator is <i>false</i> messagesInQueue is <i>true</i>	5. The CSMS sends a GetTransactionStatusRequest
	7. The OCTT sends a TransactionEventRequest With triggerReason is <i>StopAuthorized</i> eventType is <i>Updated</i> offline is <i>true</i> seqNo <This is the first of the two skipped values>	8. The CSMS responds with a TransactionEventResponse
	9. The OCTT sends a TransactionEventRequest With triggerReason is <i>ChargingStateChanged</i> transactionInfo.chargingState is <i>EVConnected</i> eventType is <i>Updated</i> offline is <i>true</i> seqNo <This is the second of the two skipped values>	10. The CSMS responds with a TransactionEventResponse
Tool validations	* Step 5: Message: GetTransactionStatusRequest - transactionId <Generated transactionId from Before> Post scenario validations: N/a	

Table 320. Test Case Id: TC_E_33_CSMS

Test case name	Check Transaction status - Without transactionId - with message in queue	
Test case Id	TC_E_33_CSMS	
Use case Id(s)	E14	
Requirement(s)	E14.FR.06,E14.FR.07	
System under test	CSMS	
Description	The CSMS is able to request the status of a transaction and to find out whether there are queued transaction-related messages, using the GetTransactionStatusRequest message.	
Purpose	To verify if the CSMS is able to request the status of queued TransactionEventRequest messages by sending a GetTransactionStatusRequest without a transactionId. The OCTT will respond that there are message(s) queued.	
Prerequisite(s)	N/a	
Before (Preparations)	<p>Configuration State: N/a</p> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>EnergyTransferStarted</i></p>	
Main (Test scenario)	<p>Charging Station</p> <p>1. The OCTT closes the WebSocket connection.</p> <p>2. The OCTT waits a number of seconds equal to _<Configured Transaction Duration>, then it will reconnect to the CSMS._</p> <p>4. The OCTT responds with a GetTransactionStatusResponse With ongoingIndicator is omitted. messagesInQueue is true</p> <p>5. The OCTT sends a TransactionEventRequest With eventType is <i>Updated</i> meterValues is present. offline is true</p>	<p>CSMS</p> <p>3. The CSMS sends a GetTransactionStatusRequest</p> <p>6. The CSMS responds with a TransactionEventResponse</p>
Tool validations	<p>* Step 3: Message: GetTransactionStatusRequest - transactionId must be omitted.</p> <p>Post scenario validations: N/a</p>	

Table 321. Test Case Id: TC_E_34_CSMS

Test case name	Check Transaction status - Without transactionId - without message in queue	
Test case Id	TC_E_34_CSMS	
Use case Id(s)	E14	
Requirement(s)	E14.FR.06,E14.FR.08	
System under test	CSMS	
Description	The CSMS is able to request the status of a transaction and to find out whether there are queued transaction-related messages, using the GetTransactionStatusRequest message.	
Purpose	To verify if the CSMS is able to request the status of queued TransactionEventRequest messages by sending a GetTransactionStatusRequest without a transactionId. The OCTT will respond that there are NO message(s) queued.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The OCTT responds with a GetTransactionStatusResponse With ongoingIndicator is omitted. messagesInQueue is <i>false</i>	1. The CSMS sends a GetTransactionStatusRequest
Tool validations	<ul style="list-style-type: none"> * Step 1: Message: GetTransactionStatusRequest - transactionId must be omitted. 	
	Post scenario validations: N/a	

Table 322. Test Case Id: TC_E_53_CSMS

Test case name	Reset Sequence Number - CSMS accepting seqNo = 0 at start of transaction	
Test case Id	TC_E_53_CSMS	
Use case Id(s)	E01	
Requirement(s)	E01.FR.07	
System under test	CSMS	
Description	OCPP 2.0.1 Edition 2 recommends that seqNo starts at 0 for every transaction. CSMS must therefore be robust to a seqNo that is not continuously increasing, but that restarts for new transactions. Since a TransactionEventRequest cannot be rejected, this can only be detected by either the complete absence of a TransactionEventResponse from CSMS or an otherwise misbehaving CSMS.	
Purpose	To verify if the CSMS accepts that a new transaction starts with a seqNo = 0.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	1. Execute Reusable State <i>EnergyTransferStarted</i> <u>Note(s):</u> New transaction will use seqNo 0 for the first TransactionEventRequest.	
	2. Execute Reusable State <i>EVDisconnected</i>	
	3. Execute Reusable State <i>EnergyTransferStarted</i> <u>Note(s):</u> New transaction will use seqNo 0 for the first TransactionEventRequest.	
	4. Execute Reusable State <i>EVDisconnected</i>	
Tool validations	* Step 1: CSMS accepts the message TransactionEventRequest with eventType = Started and seqNo = 0 and answers with a TransactionEventResponse message. * Step 3: CSMS accepts the message TransactionEventRequest with eventType = Started and seqNo = 0 and answers with a TransactionEventResponse message.	

3.7. F Remote Control

Table 323. Test Case Id: TC_F_01_CSMS

Test case name	Remote start transaction - Cable plugin first	
Test case Id	TC_F_01_CSMS	
Use case Id(s)	F01	
Requirement(s)	N/a	
System under test	CSMS	
Description	OCPP 2.x.x allows an EV driver to either first connect the EV and EVSE OR wait for/trigger a RequestStartTransactionRequest. Both sequences will result in being able to charge.	
Purpose	To verify if the CSMS is able to handle a Charging Station that starts a charging session when the EV driver first connects the EV and EVSE, before receiving a RequestStartTransactionRequest message.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): State is <i>EVConnectedPreSession</i>	
Main (Test scenario)	Charging Station	CSMS
	<u>Manual Action:</u> Trigger the CSMS to request the Charging Station to start a transaction.	
	2. The OCTT responds with a RequestStartTransactionResponse with status Accepted transactionId is <i><Generated transactionId></i>	1. The CSMS sends a RequestStartTransactionRequest
	3. The OCTT sends a TransactionEventRequest . with triggerReason is <i>RemoteStart</i> transactionInfo.remoteStartId is <i><By CSMS provided remoteStartId></i> eventType is <i>Updated</i>	4. The CSMS responds with a TransactionEventResponse .
	5. Execute Reusable State EnergyTransferStarted (State is <i>Authorized</i> and _EVConnected = true)	
Tool validations	<ul style="list-style-type: none"> * Step 1: Message: RequestStartTransactionRequest - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type> 	
	Post scenario validations: N/a	

Table 324. Test Case Id: TC_F_02_CSMS

Test case name	Remote start transaction - Remote start first - AuthorizeRemoteStart is true	
Test case Id	TC_F_02_CSMS	
Use case Id(s)	F02	
Requirement(s)	F02.FR.01, F01.FR.01	
System under test	CSMS	
Description	OCPP 2.x.x allows an EV driver to either first wait for/trigger a RequestStartTransactionRequest OR connect the EV and EVSE. Both sequences will result in being able to charge.	
Purpose	To verify if the CSMS is able to handle a Charging Station that starts a charging session when the Charging Stations receives a RequestStartTransactionRequest message (while AuthorizeRemoteStart is true), before the EV driver connects the EV and EVSE (within the connectionTimeout). The Charging Station has to authorize beforehand like a local action to start a transaction.	
Prerequisite(s)	AuthEnabled is NOT implemented with mutability ReadOnly and the value set to false	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): N/a	
Main (Test scenario)	Charging Station <u>Manual Action:</u> Trigger the CSMS to request the Charging Station to start a transaction.	CSMS
	2. The OCTT responds with a RequestStartTransactionResponse with status Accepted transactionId is omitted.	1. The CSMS sends a RequestStartTransactionRequest
	3. The OCTT sends a AuthorizeRequest . with idToken.idToken <Configured valid_idtoken_idtoken> idToken.type <Configured valid_idtoken_type>	4. The CSMS responds with a AuthorizeResponse .
	5. The OCTT sends a TransactionEventRequest . with triggerReason is <i>RemoteStart</i> transactionInfo.remoteStartId is <By OCTT generated remoteStartID> eventType is <i>Started</i>	6. The CSMS responds with a TransactionEventResponse .
	7. Execute Reusable State EnergyTransferStarted (State is <i>Authorized</i> and _EVConnected = false)	
Tool validations	<ul style="list-style-type: none"> * Step 1: Message: RequestStartTransactionRequest - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type> * Step 4: Message: AuthorizeResponse - idTokenInfo.status must be <i>Accepted</i> <p>Post scenario validations: N/a</p>	

Table 325. Test Case Id: TC_F_03_CSMS

Test case name	Remote start transaction - Remote start first - AuthorizeRemoteStart is false	
Test case Id	TC_F_03_CSMS	
Use case Id(s)	F02	
Requirement(s)	F02.FR.01, F01.FR.02	
System under test	CSMS	
Description	OCPP 2.x.x allows an EV driver to either first wait for/trigger a RequestStartTransactionRequest OR connect the EV and EVSE. Both sequences will result in being able to charge.	
Purpose	To verify if the CSMS is able to handle a Charging Station that starts a charging session when the Charging Stations receives a RequestStartTransactionRequest message (while AuthorizeRemoteStart is false), before the EV driver connects the EV and EVSE (within the connectionTimeout). The Charging station does NOT have to authorize beforehand like a local action to start a transaction.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station <u>Manual Action:</u> Trigger the CSMS to request the Charging Station to start a transaction.	CSMS
	2. The OCTT responds with a RequestStartTransactionResponse with status Accepted transactionId is omitted.	1. The CSMS sends a RequestStartTransactionRequest
	3. The OCTT sends a TransactionEventRequest . with triggerReason is RemoteStart transactionInfo.remoteStartId is <By OCTT generated remoteStartID> eventType is Started	4. The CSMS responds with a TransactionEventResponse .
	5. Execute Reusable State EnergyTransferStarted (State is Authorized and _EVConnected = false)	
Tool validations	* Step 1: Message: RequestStartTransactionRequest - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type>	
	Post scenario validations: N/a	

Table 326. Test Case Id: TC_F_04_CSMS

Test case name	Remote start transaction - Remote start first - Cable plugin timeout	
Test case Id	TC_F_04_CSMS	
Use case Id(s)	F02, E03	
Requirement(s)	E03.FR.05	
System under test	CSMS	
Description	OCPP 2.x.x allows an EV driver to either first wait for/trigger a RequestStartTransactionRequest OR connect the EV and EVSE. Both sequences will result in being able to charge.	
Purpose	To verify if the CSMS is able to handle a Charging Station that deauthorizes the transaction after the EVConnectionTimeout has been reached.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	<u>Manual Action:</u> Trigger the CSMS to request the Charging Station to start a transaction.	
	2. The OCTT responds with a RequestStartTransactionResponse with status Accepted transactionId is omitted.	1. The CSMS sends a RequestStartTransactionRequest
	3. The OCTT sends a TransactionEventRequest . with triggerReason is <i>RemoteStart</i> transactionInfo.remoteStartId is <By OCTT generated remoteStartID> eventType is <i>Started</i>	4. The CSMS responds with a TransactionEventResponse .
	5. The OCTT sends a TransactionEventRequest . with triggerReason is <i>EVConnectTimeout</i> eventType is <i>Updated</i>	6. The CSMS responds with a TransactionEventResponse .
	Note(s): - This step will be executed after the _<Configured Transaction Duration> has been reached..	
Tool validations	<ul style="list-style-type: none"> * Step 1: Message: RequestStartTransactionRequest - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type> 	
	Post scenario validations: N/a	

Table 327. Test Case Id: TC_F_06_CSMS

Test case name	Remote unlock Connector - Without ongoing transaction - Accepted	
Test case Id	TC_F_06_CSMS	
Use case Id(s)	F05	
Requirement(s)	n/a	
System under test	CSMS	
Description	This test case describes how the CSMS can be requested to sent an UnlockConnectorRequest to the charging station. It sometimes happens that a connector of a Charging Station socket does not unlock correctly. This happens most of the time when there is tension on the charging cable. This means the driver cannot unplug his charging cable from the Charging Station. To help a driver, the CSO can send a UnlockConnectorRequest to the Charging Station. The Charging Station will then try to unlock the connector again.	
Purpose	To verify if the CSMS is able to perform the remote unlock connector mechanism as described at the OCPP specification.	
Prerequisite(s)		
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The OCTT responds with a UnlockConnectorResponse with status <i>Unlocked</i>	1. The CSMS sends a UnlockConnectorRequest
Tool validations	<ul style="list-style-type: none"> * Step 1: Message UnlockConnectorRequest - evseld <Configured evseld> - connectorId <Configured connectorId> 	
	Post scenario validations: <ul style="list-style-type: none"> - N/a 	

Table 328. Test Case Id: TC_F_11_CSMS

Test case name	Trigger message - MeterValues - Specific EVSE	
Test case Id	TC_F_11_CSMS	
Use case Id(s)	F06	
Requirement(s)	F06.FR.01,F06.FR.02	
System under test	CSMS	
Description	The CSMS can request a Charging Station to send Charging Station-initiated messages. In the request the CSMS indicates which message it wishes to receive.	
Purpose	To verify if the CSMS is able to trigger the Charging Station to send a MeterValuesRequest for a specific EVSE, using a TriggerMessageRequest.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The OCTT responds with a TriggerMessageResponse with status Accepted	1. The CSMS sends a TriggerMessageRequest
	3. The OCTT sends a MeterValuesRequest With evseld <Configured evseld> meterValue[0].sampledValue.context <i>Trigger</i>	4. The CSMS responds with a MeterValuesResponse
Tool validations	<ul style="list-style-type: none"> * Step 1: Message: TriggerMessageRequest - requestedMessage must be <i>MeterValues</i> - evse.id must be <Configured evseld> 	
	Post scenario validations: N/a	

Table 329. Test Case Id: TC_F_12_CSMS

Test case name	Trigger message - MeterValues - All EVSE	
Test case Id	TC_F_12_CSMS	
Use case Id(s)	F06	
Requirement(s)	F06.FR.01	
System under test	CSMS	
Description	The CSMS can request a Charging Station to send Charging Station-initiated messages. In the request the CSMS indicates which message it wishes to receive.	
Purpose	To verify if the CSMS is able to trigger the Charging Station to send a MeterValuesRequest for all EVSE, using a TriggerMessageRequest.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): N/a	
Main (Test scenario)	Charging Station 2. The OCTT responds with a TriggerMessageResponse with status Accepted 3. The OCTT sends a MeterValuesRequest With evseld omitted meterValue[0].sampledValue.context Trigger <u>Note(s):</u> - This step will be executed for every EVSE.	CSMS 1. The CSMS sends a TriggerMessageRequest 4. The CSMS responds with a MeterValuesResponse
Tool validations	<ul style="list-style-type: none"> * Step 1: Message: TriggerMessageRequest - requestedMessage must be MeterValues <p>Post scenario validations: N/a</p>	

Table 330. Test Case Id: TC_F_13_CSMS

Test case name	Trigger message - TransactionEvent - Specific EVSE	
Test case Id	TC_F_13_CSMS	
Use case Id(s)	F06	
Requirement(s)	F06.FR.01,F06.FR.02	
System under test	CSMS	
Description	The CSMS can request a Charging Station to send Charging Station-initiated messages. In the request the CSMS indicates which message it wishes to receive.	
Purpose	To verify if the CSMS is able to trigger the Charging Station to send a TransactionEventRequest for a specific EVSE, using a TriggerMessageRequest.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): State is <i>EnergyTransferStarted</i>	
Main (Test scenario)	Charging Station 2. The OCTT responds with a TriggerMessageResponse with status Accepted	CSMS 1. The CSMS sends a TriggerMessageRequest
	3. The OCTT sends a TransactionEventRequest With evse.id <Configured evseld> triggerReason <i>Trigger</i> transactionInfo.chargingState <i>Charging</i> meterValue is present meterValue[0].sampledValue.context <i>Trigger</i>	4. The CSMS responds with a TransactionEventResponse
Tool validations	* Step 1: Message: TriggerMessageRequest - requestedMessage must be <i>TransactionEvent</i> - evse.id must be <Configured evseld> Post scenario validations: N/a	

Table 331. Test Case Id: TC_F_14_CSMS

Test case name	Trigger message - TransactionEvent - All EVSE	
Test case Id	TC_F_14_CSMS	
Use case Id(s)	F06	
Requirement(s)	F06.FR.01	
System under test	CSMS	
Description	The CSMS can request a Charging Station to send Charging Station-initiated messages. In the request the CSMS indicates which message it wishes to receive.	
Purpose	To verify if the CSMS is able to trigger the Charging Station to send a TransactionEventRequest for all EVSE, using a TriggerMessageRequest.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): State is <i>EnergyTransferStarted</i>	
Main (Test scenario)	Charging Station 2. The OCTT responds with a TriggerMessageResponse with status Accepted 3. The OCTT sends a TransactionEventRequest With evse.id omitted triggerReason <i>Trigger</i> transactionInfo.chargingState <i>Charging</i> meterValue is present meterValue[0].sampledValue.context <i>Trigger</i> <u>Note(s):</u> - This step will be executed for every EVSE.	CSMS 1. The CSMS sends a TriggerMessageRequest 4. The CSMS responds with a TransactionEventResponse
Tool validations	* Step 1: Message: TriggerMessageRequest - requestedMessage must be <i>TransactionEvent</i> Post scenario validations: N/a	

Table 332. Test Case Id: TC_F_15_CSMS

Test case name	Trigger message - LogStatusNotification - Idle	
Test case Id	TC_F_15_CSMS	
Use case Id(s)	F06	
Requirement(s)	F06.FR.01	
System under test	CSMS	
Description	The CSMS can request a Charging Station to send Charging Station-initiated messages. In the request the CSMS indicates which message it wishes to receive.	
Purpose	To verify if the CSMS is able to trigger the Charging Station to send a LogStatusNotificationRequest, using a TriggerMessageRequest.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The OCTT responds with a TriggerMessageResponse with status <i>Accepted</i>	1. The CSMS sends a TriggerMessageRequest
	3. The OCTT sends a LogStatusNotificationRequest with status Idle	4. The CSMS responds with a LogStatusNotificationResponse
Tool validations	<ul style="list-style-type: none"> * Step 1: Message: TriggerMessageRequest - requestedMessage must be <i>LogStatusNotification</i> 	
	Post scenario validations: N/a	

Table 333. Test Case Id: TC_F_18_CSMS

Test case name	Trigger message - FirmwareStatusNotification - Idle	
Test case Id	TC_F_18_CSMS	
Use case Id(s)	F06	
Requirement(s)	F06.FR.01	
System under test	CSMS	
Description	The CSMS can request a Charging Station to send Charging Station-initiated messages. In the request the CSMS indicates which message it wishes to receive.	
Purpose	To verify if the CSMS is able to trigger the Charging Station to send a FirmwareStatusNotificationRequest, using a TriggerMessageRequest.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The OCTT responds with a TriggerMessageResponse with status Accepted	1. The CSMS sends a TriggerMessageRequest
	3. The OCTT sends a FirmwareStatusNotificationRequest with status Idle	4. The CSMS responds with a FirmwareStatusNotificationResponse
Tool validations	<ul style="list-style-type: none"> * Step 1: <p>Message: TriggerMessageRequest - requestedMessage must be <i>FirmwareStatusNotification</i></p>	
	<p>Post scenario validations: N/a</p>	

Table 334. Test Case Id: TC_F_20_CSMS

Test case name	Trigger message - Heartbeat	
Test case Id	TC_F_20_CSMS	
Use case Id(s)	F06	
Requirement(s)	F06.FR.01	
System under test	CSMS	
Description	The CSMS can request a Charging Station to send Charging Station-initiated messages. In the request the CSMS indicates which message it wishes to receive.	
Purpose	To verify if the CSMS is able to trigger the Charging Station to send a HeartbeatRequest, using a TriggerMessageRequest.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The OCTT responds with a TriggerMessageResponse with status Accepted	1. The CSMS sends a TriggerMessageRequest
	3. The OCTT sends a HeartbeatRequest	4. The CSMS responds with a HeartbeatResponse
Tool validations	<ul style="list-style-type: none"> * Step 1: Message: TriggerMessageRequest - requestedMessage must be <i>Heartbeat</i> 	
	Post scenario validations: N/a	

Table 335. Test Case Id: TC_F_23_CSMS

Test case name	Trigger message - StatusNotification - Specific EVSE - Available	
Test case Id	TC_F_23_CSMS	
Use case Id(s)	F06	
Requirement(s)	F06.FR.01,F06.FR.02,F06.FR.13	
System under test	CSMS	
Description	The CSMS can request a Charging Station to send Charging Station-initiated messages. In the request the CSMS indicates which message it wishes to receive.	
Purpose	To verify if the CSMS is able to trigger the Charging Station to send a StatusNotificationRequest for a specific available EVSE, using a TriggerMessageRequest.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): N/a	
Main (Test scenario)	Charging Station 2. The OCTT responds with a TriggerMessageResponse with status Accepted 3. The OCTT notifies the CSMS about the current state of the connector. Message: StatusNotificationRequest - connectorStatus Available - evseld <Configured evseld> - connectorId <Configured connectorId> Message: NotifyEventRequest - trigger Delta - actualValue "Available" - component.name "Connector" - component.evse.id <Configured evseld> - component.evse.connectorid <Configured connectorId> - variable.name "AvailabilityState"	CSMS 1. The CSMS sends a TriggerMessageRequest 4. The CSMS responds accordingly.
Tool validations	* Step 1: Message: TriggerMessageRequest - requestedMessage must be StatusNotification - evse.id must be <Configured evseld> Post scenario validations: N/a	

Table 336. Test Case Id: TC_F_24_CSMS

Test case name	Trigger message - StatusNotification - Specific EVSE - Occupied	
Test case Id	TC_F_24_CSMS	
Use case Id(s)	F06	
Requirement(s)	F06.FR.01,F06.FR.02,F06.FR.13	
System under test	CSMS	
Description	The CSMS can request a Charging Station to send Charging Station-initiated messages. In the request the CSMS indicates which message it wishes to receive.	
Purpose	To verify if the CSMS is able to trigger the Charging Station to send a StatusNotificationRequest for a specific occupied EVSE, using a TriggerMessageRequest.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): N/a	
Main (Test scenario)	Charging Station 1. The OCTT notifies the CSMS about the current state of the connector. Message: StatusNotificationRequest - connectorStatus Occupied - evseld <Configured evseld> - connectorId <Configured connectorId> Message: NotifyEventRequest - trigger Delta - actualValue "Occupied" - component.name "Connector" - component.evse.id <Configured evseld> - component.evse.connectorid <Configured connectorId> - variable.name "AvailabilityState"	CSMS 2. The CSMS responds accordingly.
	 4. The OCTT responds with a TriggerMessageResponse with status Accepted	 3. The CSMS sends a TriggerMessageRequest
	 5. The OCTT notifies the CSMS about the current state of the connector. Message: StatusNotificationRequest - connectorStatus Occupied - evseld <Configured evseld> - connectorId <Configured connectorId> Message: NotifyEventRequest - trigger Delta - actualValue "Occupied" - component.name "Connector" - component.evse.id <Configured evseld> - component.evse.connectorid <Configured connectorId> - variable.name "AvailabilityState"	 6. The CSMS responds accordingly.

Test case name	Trigger message - StatusNotification - Specific EVSE - Occupied
Tool validations	<p>* Step 1:</p> <p>Message: TriggerMessageRequest</p> <ul style="list-style-type: none">- requestedMessage must be <i>StatusNotification</i>- evse.id must be <<i>Configured evsId</i>> <p>Post scenario validations:</p> <p>N/a</p>

Table 337. Test Case Id: TC_F_27_CSMS

Test case name	Trigger message - NotImplemented	
Test case Id	TC_F_27_CSMS	
Use case Id(s)	F06	
Requirement(s)	F06.FR.08	
System under test	CSMS	
Description	The CSMS can request a Charging Station to send Charging Station-initiated messages. In the request the CSMS indicates which message it wishes to receive.	
Purpose	To verify if the CSMS is able to handle a Charging Station that does not support the requested message value from a TriggerMessageRequest.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The OCTT responds with a TriggerMessageResponse with status <i>NotImplemented</i>	1. The CSMS sends a TriggerMessageRequest
Tool validations	N/a	
	Post scenario validations: N/a	

3.8. G Availability

Table 338. Test Case Id: TC_G_03_CSMS

Test case name	Change Availability EVSE - Operative to inoperative	
Test case Id	TC_G_03_CSMS	
Use case Id(s)	G03	
Requirement(s)	N/a	
System under test	CSMS	
Description	This test case covers how the CSMS requests the Charging Station to change the availability of one of the EVSEs from Operative to Inoperative. An EVSE is considered Operative in any status other than Faulted and Unavailable.	
Purpose	To verify if the CSMS is able to perform the change availability mechanism as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	1. Execute Reusable State <i>Unavailable</i> for <Configured evsId>	
Tool validations	N/a	
	Post scenario validations: - N/a	

Table 339. Test Case Id: TC_G_04_CSMS

Test case name	Change Availability EVSE - Inoperative to operative	
Test case Id	TC_G_04_CSMS	
Use case Id(s)	G03	
Requirement(s)	N/a	
System under test	CSMS	
Description	This test case covers how the CSMS requests the Charging Station to change the availability of one of the EVSEs from Inoperative to Operative. An EVSE is considered Operative in any status other than Faulted and Unavailable.	
Purpose	To verify if the CSMS is able to perform the change availability mechanism as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	<p>Configuration State: N/a</p> <p>Memory State: <i>Unavailable</i> for <Configured evseld></p> <p>Reusable State(s): N/a</p>	
Main (Test scenario)	<p>Charging Station</p> <p>Manual Action: Request the CSMS to change the availability of an EVSE to Operative.</p> <p>2. The OCTT responds with a ChangeAvailabilityResponse with status Accepted</p> <p>3. The OCTT notifies the CSMS about the current state of all connectors belonging to the specified EVSE (and optionally also from the EVSE itself). Message: StatusNotificationRequest - connectorStatus Available - evseld <Configured evseld> Message: NotifyEventRequest - trigger Delta - actualValue "Available" - component.name "EVSE" / Connector - component.evse.id <Configured evseld> - variable.name "AvailabilityState"</p>	<p>1. The CSMS sends a ChangeAvailabilityRequest</p> <p>4. The CSMS responds accordingly.</p>
Tool validations	<p>* Step 1: Message ChangeAvailabilityRequest - operationalStatus Operative - evse.id <Configured evseld> - connectorId omit</p> <p>Post scenario validations: - N/a</p>	

Table 340. Test Case Id: TC_G_05_CSMS

Test case name	Change Availability Charging Station - Operative to inoperative									
Test case Id	TC_G_05_CSMS									
Use case Id(s)	G04									
Requirement(s)	N/a									
System under test	CSMS									
Description	<p>This test case describes how the CSMS requests the Charging Station to change the availability from operative to inoperative.</p> <p>A Charging Station is considered Operative when it is charging or ready for charging.</p> <p>A Charging Station is considered Inoperative when it does not allow any charging.</p>									
Purpose	To verify if the CSMS is able to perform the change availability mechanism as described at the OCPP specification.									
Prerequisite(s)	n/a									
Before (Preparations)	<p>Configuration State: N/a</p> <p>Memory State: N/a</p> <p>Reusable State(s): N/a</p>									
Main (Test scenario)	<table border="1"> <tr> <td>Charging Station</td> <td>CSMS</td> </tr> <tr> <td colspan="2"><u>Manual Action: Request the CSMS to change the availability of the Charging Station to Inoperative.</u></td> </tr> <tr> <td>2. The OCTT responds with a ChangeAvailabilityResponse with status Accepted</td> <td>1. The CSMS sends a ChangeAvailabilityRequest</td> </tr> <tr> <td>3. The OCTT notifies the CSMS about the current state of all connectors Message: StatusNotificationRequest - connectorStatus Unavailable Message: NotifyEventRequest - trigger Delta - actualValue "Unavailable" - component.name "Connector" - variable.name "AvailabilityState"</td> <td>4. The CSMS responds accordingly.</td> </tr> </table>		Charging Station	CSMS	<u>Manual Action: Request the CSMS to change the availability of the Charging Station to Inoperative.</u>		2. The OCTT responds with a ChangeAvailabilityResponse with status Accepted	1. The CSMS sends a ChangeAvailabilityRequest	3. The OCTT notifies the CSMS about the current state of all connectors Message: StatusNotificationRequest - connectorStatus Unavailable Message: NotifyEventRequest - trigger Delta - actualValue "Unavailable" - component.name "Connector" - variable.name "AvailabilityState"	4. The CSMS responds accordingly.
Charging Station	CSMS									
<u>Manual Action: Request the CSMS to change the availability of the Charging Station to Inoperative.</u>										
2. The OCTT responds with a ChangeAvailabilityResponse with status Accepted	1. The CSMS sends a ChangeAvailabilityRequest									
3. The OCTT notifies the CSMS about the current state of all connectors Message: StatusNotificationRequest - connectorStatus Unavailable Message: NotifyEventRequest - trigger Delta - actualValue "Unavailable" - component.name "Connector" - variable.name "AvailabilityState"	4. The CSMS responds accordingly.									
Tool validations	<p>* Step 1: Message ChangeAvailabilityRequest - operationalStatus Inoperative - evseld omit - connectordId omit</p> <p>Post scenario validations: - N/a</p>									

Table 341. Test Case Id: TC_G_06_CSMS

Test case name	Change Availability Charging Station - Inoperative to operative									
Test case Id	TC_G_06_CSMS									
Use case Id(s)	G04									
Requirement(s)	N/a									
System under test	CSMS									
Description	<p>This test case describes how the CSMS requests the Charging Station to change the availability from inoperative to operative.</p> <p>A Charging Station is considered Operative when it is charging or ready for charging.</p> <p>A Charging Station is considered Inoperative when it does not allow any charging.</p>									
Purpose	To verify if the CSMS is able to perform the change availability mechanism as described at the OCPP specification.									
Prerequisite(s)	n/a									
Before (Preparations)	<p>Configuration State: N/a</p> <p>Memory State: N/a</p> <p>Reusable State(s): Charging Station set to <i>Unavailable</i> (Original status was Available)</p>									
Main (Test scenario)	<table border="1"> <thead> <tr> <th>Charging Station</th> <th>CSMS</th> </tr> </thead> <tbody> <tr> <td><u>Manual Action: Request the CSMS to change the availability of the Charging Station to Inoperative.</u></td><td>1. The CSMS sends a ChangeAvailabilityRequest</td></tr> <tr> <td>2. The OCTT responds with a ChangeAvailabilityResponse with status Accepted</td><td></td></tr> <tr> <td>3. The OCTT notifies the CSMS about the current state of all connectors. Message: StatusNotificationRequest - connectorStatus Available Message: NotifyEventRequest - trigger Delta - actualValue "Available" - component.name "Connector" - variable.name "AvailabilityState"</td><td>4. The CSMS responds accordingly.</td></tr> </tbody> </table>		Charging Station	CSMS	<u>Manual Action: Request the CSMS to change the availability of the Charging Station to Inoperative.</u>	1. The CSMS sends a ChangeAvailabilityRequest	2. The OCTT responds with a ChangeAvailabilityResponse with status Accepted		3. The OCTT notifies the CSMS about the current state of all connectors. Message: StatusNotificationRequest - connectorStatus Available Message: NotifyEventRequest - trigger Delta - actualValue "Available" - component.name "Connector" - variable.name "AvailabilityState"	4. The CSMS responds accordingly.
Charging Station	CSMS									
<u>Manual Action: Request the CSMS to change the availability of the Charging Station to Inoperative.</u>	1. The CSMS sends a ChangeAvailabilityRequest									
2. The OCTT responds with a ChangeAvailabilityResponse with status Accepted										
3. The OCTT notifies the CSMS about the current state of all connectors. Message: StatusNotificationRequest - connectorStatus Available Message: NotifyEventRequest - trigger Delta - actualValue "Available" - component.name "Connector" - variable.name "AvailabilityState"	4. The CSMS responds accordingly.									
Tool validations	<p>* Step 1: Message ChangeAvailabilityRequest - operationalStatus Operative - evseld omit - connectordId omit</p> <p>Post scenario validations: - N/a</p>									

Table 342. Test Case Id: TC_G_07_CSMS

Test case name	Change Availability Connector - Operative to inoperative	
Test case Id	TC_G_07_CSMS	
Use case Id(s)	G03	
Requirement(s)	N/a	
System under test	CSMS	
Description	This test case covers how the CSMS requests the Charging Station to change the availability of one of the Connectors of one EVSE from Operative to Inoperative. A Connector is considered Operative in any status other than Faulted and Unavailable.	
Purpose	To verify if the CSMS is able to perform the change availability mechanism as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): N/a	
Main (Test scenario)	Charging Station <u>Manual Action:</u> Request the CSMS to change the availability of a Connector to Inoperative. 2. The OCTT responds with a ChangeAvailabilityResponse with status Accepted 3. The OCTT notifies the CSMS about the current state of the connector. Message: StatusNotificationRequest - connectorStatus <i>Unavailable</i> - evseld <Configured evseld> - connectorId <Configured connectorId> Message: NotifyEventRequest - trigger <i>Delta</i> - actualValue "Unavailable" - component.name "Connector" - component.evse.id <Configured evseld> - component.evse.connectorid <Configured connectorId> - variable.name "AvailabilityState"	CSMS 1. The CSMS sends a ChangeAvailabilityRequest 4. The CSMS responds accordingly.
Tool validations	* Step 1: Message ChangeAvailabilityRequest - operationalStatus <i>Inoperative</i> - evse.id <Configured evseld> - evse.connectorId <Configured connectorId> Post scenario validations: N/a	

Table 343. Test Case Id: TC_G_08_CSMS

Test case name	Change Availability Connector - Inoperative to operative	
Test case Id	TC_G_08_CSMS	
Use case Id(s)	G03	
Requirement(s)	N/a	
System under test	CSMS	
Description	This test case covers how the CSMS requests the Charging Station to change the availability of one of the Connectors from one EVSE from Inoperative to Operative. A Connector is considered Operative in any status other than Faulted and Unavailable.	
Purpose	To verify if the CSMS is able to perform the change availability mechanism as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	<p>Configuration State: N/a</p> <p>Memory State: <i>Unavailable</i> for <Configured connectorId></p> <p>Reusable State(s): N/a</p>	
Main (Test scenario)	<p>Charging Station</p> <p><u>Manual Action:</u> Request the CSMS to change the availability of a Connector to Operative.</p> <p>2. The OCTT responds with a ChangeAvailabilityResponse with status Accepted</p> <p>3. The OCTT notifies the CSMS about the current state of the connector. Message: StatusNotificationRequest - connectorStatus Available - evselId <Configured evselId> - connectorId <Configured connectorId></p> <p>Message: NotifyEventRequest - trigger Delta - actualValue "Available" - component.name "Connector" - component.evse.id <Configured evselId> - component.evse.connectorid <Configured connectorId> - variable.name "AvailabilityState"</p>	<p>1. The CSMS sends a ChangeAvailabilityRequest</p> <p>4. The CSMS responds accordingly.</p>
Tool validations	<p>* Step 1: Message ChangeAvailabilityRequest - operationalStatus Operative - evse.id <Configured evselId> - evse.connectorId <Configured connectorId></p> <p>Post scenario validations: N/a</p>	

Table 344. Test Case Id: TC_G_11_CSMS

Test case name	Change Availability EVSE - With ongoing transaction	
Test case Id	TC_G_11_CSMS	
Use case Id(s)	G03	
Requirement(s)	N/a	
System under test	CSMS	
Description	This test case covers how the CSMS requests the Charging Station to change the availability of one of the EVSEs from Operative to Inoperative. An EVSE is considered Operative in any status other than Faulted and Unavailable.	
Purpose	To verify if the CSMS is able to send a change availability request during a transaction according to the mechanism as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	<p>Configuration State: N/a</p> <p>Memory State: N/a</p> <p>Reusable State: State is <i>EnergyTransferStarted</i></p>	
Main (Test scenario)	<p>Charging Station</p> <p><u>Note(s):</u> Request the CSMS to change the availability to inoperative</p> <p>2. The OCTT responds with a ChangeAvailabilityResponse with status Scheduled</p> <p><u>Note(s):</u> Wait for <Configured Transaction Duration></p> <p>3. Execute Reusable State StopAuthorized</p> <p>4. Execute Reusable State EVConnectedPostSession</p> <p>5. Execute Reusable State EVDisconnected</p> <p>6. The OCTT notifies the CSMS about the current state of all connectors with Message: StatusNotificationRequest - connectorStatus Unavailable - evselid <Configured evselid> OR Message: NotifyEventRequest - trigger Delta - actualValue "Unavailable" - component.name "Connector" - component.evse.id <Configured evselid> - variable.name "AvailabilityState"</p>	<p>CSMS</p> <p>1. The CSMS sends a ChangeAvailabilityRequest</p> <p>7. The CSMS responds accordingly.</p>
Tool validations	<ul style="list-style-type: none"> * Step 1: Message ChangeAvailabilityRequest - operationalStatus Inoperative - evse.id <Configured evselid> - connectorId omit Post scenario validations: - A respond to report the state of a connector has been received for all connectors. 	

Table 345. Test Case Id: TC_G_14_CSMS

Test case name	Change Availability Charging Station - With ongoing transaction																			
Test case Id	TC_G_14_CSMS																			
Use case Id(s)	G04																			
Requirement(s)	N/a																			
System under test	CSMS																			
Description	This test case covers how the CSMS requests the Charging Station to change the availability of one of the EVSEs from Operative to Inoperative. An EVSE is considered Operative in any status other than Faulted and Unavailable.																			
Purpose	To verify if the CSMS is able to send a change availability request during a transaction according to the mechanism as described at the OCPP specification.																			
Prerequisite(s)	n/a																			
Before (Preparations)	<p>Configuration State: N/a</p> <p>Memory State: N/a</p> <p>Reusable State: State is <i>EnergyTransferStarted</i></p>																			
Main (Test scenario)	<table border="1"> <tr> <td>Charging Station</td> <td>CSMS</td> </tr> <tr> <td><u>Note(s):</u> Request the CSMS to change the availability of the station to inoperative</td> <td>1. The CSMS sends a ChangeAvailabilityRequest</td> </tr> <tr> <td>2. The OCTT responds with a ChangeAvailabilityResponse with status Scheduled</td> <td></td> </tr> <tr> <td>3. The OCTT notifies the CSMS about the current state of all unoccupied connectors with Message: StatusNotificationRequest - connectorStatus Unavailable</td> <td>4. The CSMS responds accordingly.</td> </tr> <tr> <td><u>Note(s):</u> Wait for <Configured Transaction Duration></td> <td></td> </tr> <tr> <td>5. Execute Reusable State StopAuthorized</td> <td></td> </tr> <tr> <td>6. Execute Reusable State EVConnectedPostSession</td> <td></td> </tr> <tr> <td>7. Execute Reusable State EVDDisconnected</td> <td></td> </tr> <tr> <td>8. The OCTT notifies the CSMS about the current state of the configured connector with Message: StatusNotificationRequest - connectorStatus Unavailable</td> <td>9. The CSMS responds accordingly.</td> </tr> </table>	Charging Station	CSMS	<u>Note(s):</u> Request the CSMS to change the availability of the station to inoperative	1. The CSMS sends a ChangeAvailabilityRequest	2. The OCTT responds with a ChangeAvailabilityResponse with status Scheduled		3. The OCTT notifies the CSMS about the current state of all unoccupied connectors with Message: StatusNotificationRequest - connectorStatus Unavailable	4. The CSMS responds accordingly.	<u>Note(s):</u> Wait for <Configured Transaction Duration>		5. Execute Reusable State StopAuthorized		6. Execute Reusable State EVConnectedPostSession		7. Execute Reusable State EVDDisconnected		8. The OCTT notifies the CSMS about the current state of the configured connector with Message: StatusNotificationRequest - connectorStatus Unavailable	9. The CSMS responds accordingly.	
Charging Station	CSMS																			
<u>Note(s):</u> Request the CSMS to change the availability of the station to inoperative	1. The CSMS sends a ChangeAvailabilityRequest																			
2. The OCTT responds with a ChangeAvailabilityResponse with status Scheduled																				
3. The OCTT notifies the CSMS about the current state of all unoccupied connectors with Message: StatusNotificationRequest - connectorStatus Unavailable	4. The CSMS responds accordingly.																			
<u>Note(s):</u> Wait for <Configured Transaction Duration>																				
5. Execute Reusable State StopAuthorized																				
6. Execute Reusable State EVConnectedPostSession																				
7. Execute Reusable State EVDDisconnected																				
8. The OCTT notifies the CSMS about the current state of the configured connector with Message: StatusNotificationRequest - connectorStatus Unavailable	9. The CSMS responds accordingly.																			
Tool validations	<ul style="list-style-type: none"> * Step 1: Message ChangeAvailabilityRequest - operationalStatus Inoperative - evsId omit - connectorId omit Post scenario validations: - A respond to report the state of a connector has been received for all connectors. 																			

Table 346. Test Case Id: TC_G_17_CSMS

Test case name	Change Availability Connector - With ongoing transaction	
Test case Id	TC_G_17_CSMS	
Use case Id(s)	G03	
Requirement(s)	N/a	
System under test	CSMS	
Description	This test case covers how the CSMS requests the Charging Station to change the availability of one of the EVSEs from Operative to Inoperative. An EVSE is considered Operative in any status other than Faulted and Unavailable.	
Purpose	To verify if the CSMS is able to send a change availability request during a transaction according to the mechanism as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State: State is <i>EnergyTransferStarted</i>	
Main (Test scenario)	Charging Station <u>Note(s):</u> Request the CSMS to change the availability of one connector to inoperative	CSMS <u>1.</u> The CSMS sends a ChangeAvailabilityRequest <u>2.</u> The OCTT responds with a ChangeAvailabilityResponse with status Scheduled <u>Note(s):</u> Wait for <Configured Transaction Duration> <u>3.</u> Execute Reusable State StopAuthorized <u>4.</u> Execute Reusable State EVConnectedPostSession <u>5.</u> Execute Reusable State EVDisconnected <u>6.</u> The OCTT notifies the CSMS about the current state of all connectors with Message: StatusNotificationRequest - connectorStatus <i>Unavailable</i> - evselid <Configured evselid> - connectorId <Configured connectorId>
		<u>7.</u> The CSMS responds accordingly.
Tool validations	<ul style="list-style-type: none"> * Step 1: Message ChangeAvailabilityRequest - operationalStatus <i>Inoperative</i> - evse.id <Configured evselid> - evse.connectorId <Configured connectorId> <p>Post scenario validations:</p> <ul style="list-style-type: none"> - A respond to report the state of a connector has been received for all connectors. 	

Table 347. Test Case Id: TC_G_20_CSMS

Test case name	Connector status Notification - Lock Failure	
Test case Id	TC_G_20_CSMS	
Use case Id(s)	G05	
Requirement(s)	G05.FR.03	
System under test	CSMS	
Description	This test case describes how the EV Driver is prevented from starting a charge session at the Charging Station while the Connector is not locked properly.	
Purpose	To verify if the CSMS responds on a notifyeventrequest as described at the OCPP specification.	
Prerequisite(s)	- N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	Charging Station	CSMS
	1. The OCTT sends a NotifyEventRequest with - eventData.trigger <i>Delta</i> - eventData.component.name "ConnectorPlugRetentionLock" - eventData.variable.name "Problem" - eventData.actualValue "true"	2. The CSMS responds with a NotifyEventResponse
Tool validations	N/a	
	Post scenario validations: - N/a	

3.9. H Reservation

This section is intentionally blank, this will be added in a later version.

3.10. I Tariff and Cost

This section is intentionally blank, this will be added in a later version.

3.11. J MeterValues

Table 348. Test Case Id: TC_J_01_CSMS

Test case name	Clock-aligned Meter Values - No transaction ongoing	
Test case Id	TC_J_01_CSMS	
Use case Id(s)	J01	
Requirement(s)	J01.FR.18	
System under test	CSMS	
Description	The Charging Station samples the electrical meter or other sensor/transducer hardware to provide information about its Meter Values. Depending on configuration settings, the Charging Station will send Meter Values.	
Purpose	To verify if the CSMS is able to handle a Charging Station sending clock-aligned Meter Values, when there is no ongoing transaction.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station 1. The OCTT notifies the CSMS about its measured Meter Values. Message: MeterValuesRequest - timestamp <The intervals between the timestamps of the received Meter Value messages equals the _<Configured clock-aligned Meter Values interval>. - sampledValue.context is Sample.Clock Message: NotifyEventRequest - timestamp <The intervals between the timestamps of the received Meter Value messages equals the _<Configured clock-aligned Meter Values interval>. - trigger is Periodic - component.name is FiscalMetering Note(s): - This step will be executed every _<Configured clock-aligned Meter Values interval> - This step will be executed for evseld=0 and all configured EVSE. - The OCTT will end the testcase after it has send three Meter Value messages.	CSMS 2. The CSMS responds accordingly.
Tool validations	N/a	
	Post scenario validations: N/a	

Table 349. Test Case Id: TC_J_02_CSMS

Test case name	Clock-aligned Meter Values - Transaction ongoing	
Test case Id	TC_J_02_CSMS	
Use case Id(s)	J01	
Requirement(s)	J01.FR.18	
System under test	CSMS	
Description	The Charging Station samples the electrical meter or other sensor/transducer hardware to provide information about its Meter Values. Depending on configuration settings, the Charging Station will send Meter Values.	
Purpose	To verify if the CSMS is able to handle a Charging Station sending clock-aligned Meter Values, when there is an ongoing transaction.	
Prerequisite(s)	N/a	
Before (Preparations)	<p>Configuration State: N/a</p> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>EnergyTransferStarted</i> for <Configured evselid></p>	
Main (Test scenario)	<p>Charging Station</p> <p>1. The OCTT notifies the CSMS about its measured Meter Values.</p> <p>Message: MeterValuesRequest - timestamp <The intervals between the timestamps of the received Meter Value messages equals the configured clock-aligned Meter Values interval> - sampledValue.context is Sample.Clock</p> <p>Message: NotifyEventRequest - timestamp <The intervals between the timestamps of the received Meter Value messages equals the configured clock-aligned Meter Values interval> - trigger is Periodic - component.name is FiscalMetering</p> <p>Note(s): - This step will be executed every _<Configured clock-aligned Meter Values interval> - This step will be executed for evselid=0 and all configured idle EVSE.</p> <p>3. The OCTT sends a TransactionEventRequest With triggerReason is MeterValueClock eventType is Updated timestamp <The intervals between the timestamps of the received Meter Value messages equals the configured clock-aligned Meter Values interval> sampledValue.context is Sample.Clock</p> <p>Note(s): - This step will be executed every _<Configured clock-aligned Meter Values interval> - The OCTT will end the testcase after the _<Configured transaction duration> is reached...</p>	<p>CSMS</p> <p>2. The CSMS responds accordingly.</p> <p>4. The CSMS responds with a TransactionEventResponse</p>

Test case name	Clock-aligned Meter Values - Transaction ongoing
Tool validations	N/a
	Post scenario validations: N/a

Table 350. Test Case Id: TC_J_03_CSMS

Test case name	Clock-aligned Meter Values - EventType Ended	
Test case Id	TC_J_03_CSMS	
Use case Id(s)	J01	
Requirement(s)	J01.FR.18	
System under test	CSMS	
Description	The Charging Station samples the electrical meter or other sensor/transducer hardware to provide information about its Meter Values. Depending on configuration settings, the Charging Station will send Meter Values.	
Purpose	To verify if the CSMS is able to handle a Charging Station sending clock-aligned Meter Values, when a transaction ends.	
Prerequisite(s)	N/a	
Before (Preparations)	<p>Configuration State: N/a</p> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>EnergyTransferStarted</i></p>	
Main (Test scenario)	<p>Charging Station</p> <p>1. Execute Reusable State <i>EVDisconnected</i></p> <ul style="list-style-type: none"> - The TransactionEventRequest containing eventType <i>Ended</i> contains the MeterValue field. - timestamp <The intervals between the timestamps of the received Meter Value messages equals the configured value at configured clock-aligned Tx ended Meter Values interval>. - sampledValue.context is Sample.Clock AND the last one has Transaction.End <p>Note(s):</p> <ul style="list-style-type: none"> - This step will be executed after the _<Configured transaction duration> is reached._ - This causes the transaction to stop. 	CSMS
Tool validations	<p>N/a</p> <p>Post scenario validations: N/a</p>	

Table 351. Test Case Id: TC_J_04_CSMS

Test case name	Clock-aligned Meter Values - Signed	
Test case Id	TC_J_04_CSMS	
Use case Id(s)	J01	
Requirement(s)	J01.FR.21	
System under test	CSMS	
Description	The Charging Station samples the electrical meter or other sensor/transducer hardware to provide information about its Meter Values. Depending on configuration settings, the Charging Station will send Meter Values.	
Purpose	To verify if the CSMS is able to handle a Charging Station sending clock-aligned Meter Values, when a transaction ends.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): State is <i>EnergyTransferStarted</i>	
Main (Test scenario)	Charging Station	CSMS
	<p>1. Execute Reusable State <i>EVDisconnected</i></p> <ul style="list-style-type: none"> - The TransactionEventRequest containing eventType <i>Ended</i> contains the MeterValue field. - timestamp <The intervals between the timestamps of the received Meter Value messages equals the configured value at configured clock-aligned Tx ended Meter Values interval>. - sampledValue.context is <i>Sample.Clock</i> AND the last one has <i>Transaction.End</i> - sampledValue.signedMeterValue is <Generated SignedMeterValueType> <p>Note(s):</p> <ul style="list-style-type: none"> - This step will be executed after the <Configured transaction duration> is reached._ - This causes the transaction to stop. 	
Tool validations	N/a	
	Post scenario validations: N/a	

Table 352. Test Case Id: TC_J_07_CSMS

Test case name	Sampled Meter Values - EventType Started - EVSE known	
Test case Id	TC_J_07_CSMS	
Use case Id(s)	J02	
Requirement(s)	J02.FR.19	
System under test	CSMS	
Description	The Charging Station samples the electrical meter or other sensor/transducer hardware to provide information about its Meter Values. Depending on configuration settings, the Charging Station will send Meter Values.	
Purpose	To verify if the CSMS is able to handle a Charging Station sending start sampled Meter Values, when a transaction starts.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	1. Execute Reusable State <i>EVConnectedPreSession</i> - The TransactionEventRequest contains the MeterValue field. - sampledValue.context is <i>Transaction.Begin</i>	
Tool validations	N/a	
	Post scenario validations: N/a	

Table 353. Test Case Id: TC_J_08_CSMS

Test case name	Sampled Meter Values - Context Transaction.Begin - EVSE not known	
Test case Id	TC_J_08_CSMS	
Use case Id(s)	J02	
Requirement(s)	J02.FR.19	
System under test	CSMS	
Description	The Charging Station samples the electrical meter or other sensor/transducer hardware to provide information about its Meter Values. Depending on configuration settings, the Charging Station will send Meter Values.	
Purpose	To verify if the CSMS is able to handle a Charging Station sending start sampled Meter Values, when a transaction starts.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	1. Execute Reusable State <i>Authorized</i>	
	2. Execute Reusable State <i>EVConnectedPreSession</i>	
	<ul style="list-style-type: none"> - The TransactionEventRequest contains the MeterValue field. - sampledValue.context is <i>Transaction.Begin</i> 	
Tool validations	3. Execute Reusable State <i>EnergyTransferStarted</i>	
	N/a	
Post scenario validations: N/a		

Table 354. Test Case Id: TC_J_09_CSMS

Test case name	Sampled Meter Values - EventType Updated	
Test case Id	TC_J_09_CSMS	
Use case Id(s)	J02	
Requirement(s)	J02.FR.19	
System under test	CSMS	
Description	The Charging Station samples the electrical meter or other sensor/transducer hardware to provide information about its Meter Values. Depending on configuration settings, the Charging Station will send Meter Values.	
Purpose	To verify if the CSMS is able to handle a Charging Station sending sampled Meter Values, when there is an ongoing transaction.	
Prerequisite(s)	N/a	
Before (Preparations)	<p>Configuration State: N/a</p> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>EnergyTransferStarted</i></p>	
Main (Test scenario)	<p>Charging Station</p> <p>1. The OCTT sends a TransactionEventRequest With triggerReason is <i>MeterValuePeriodic</i> eventType is <i>Updated</i> timestamp <The intervals between the timestamps of the received Meter Value messages equals the configured sampled Meter Values interval>. sampledValue.context is <i>Sample.Periodic</i></p> <p>Note(s): - This step will be executed every <Configured sampled Meter Values interval> - The OCTT will end the testcase after three MeterValues.</p>	<p>CSMS</p> <p>2. The CSMS responds with a TransactionEventResponse</p>
Tool validations	<p>N/a</p> <p>Post scenario validations: N/a</p>	

Table 355. Test Case Id: TC_J_10_CSMS

Test case name	Sampled Meter Values - EventType Ended	
Test case Id	TC_J_10_CSMS	
Use case Id(s)	J02	
Requirement(s)	J02.FR.19	
System under test	CSMS	
Description	The Charging Station samples the electrical meter or other sensor/transducer hardware to provide information about its Meter Values. Depending on configuration settings, the Charging Station will send Meter Values.	
Purpose	To verify if the CSMS is able to handle a Charging Station sending sampled Meter Values, when a transaction ends.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): State is <i>EnergyTransferStarted</i>	
Main (Test scenario)	Charging Station	CSMS
	<p>1. Execute Reusable State <i>EVDisconnected</i></p> <ul style="list-style-type: none"> - The TransactionEventRequest containing eventType <i>Ended</i> contains the MeterValue field. - timestamp <The intervals between the timestamps of the received Meter Value messages equals the configured value at configured clock-aligned Tx ended Meter Values interval>. - sampledValue.context is Sample.Periodic AND the last one has Transaction.End <p>Note(s):</p> <ul style="list-style-type: none"> - This step will be executed after the _<Configured transaction duration> is reached._ - This causes the transaction to stop. 	
Tool validations	N/a	
	Post scenario validations: N/a	

Table 356. Test Case Id: TC_J_11_CSMS

Test case name	Sampled Meter Values - Signed	
Test case Id	TC_J_11_CSMS	
Use case Id(s)	J02	
Requirement(s)	J02.FR.21	
System under test	CSMS	
Description	The Charging Station samples the electrical meter or other sensor/transducer hardware to provide information about its Meter Values. Depending on configuration settings, the Charging Station will send Meter Values.	
Purpose	To verify if the CSMS is able to handle a Charging Station sending sampled Meter Values, when a transaction ends.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): State is <i>EnergyTransferStarted</i>	
Main (Test scenario)	Charging Station	CSMS
	<p>1. Execute Reusable State <i>EVDisconnected</i></p> <ul style="list-style-type: none"> - The TransactionEventRequest containing eventType <i>Ended</i> contains the MeterValue field. - timestamp <The intervals between the timestamps of the received Meter Value messages equals the configured value at configured clock-aligned Tx ended Meter Values interval>. - sampledValue.context is Sample.Periodic AND the last one has Transaction.End - sampledValue.signedMeterValue is <Generated SignedMeterValueType> <p>Note(s):</p> <ul style="list-style-type: none"> - This step will be executed after the <Configured transaction duration> is reached._ - This causes the transaction to stop. 	
Tool validations	N/a	
	Post scenario validations: N/a	

3.12. K SmartCharging

This section is intentionally blank, this will be added in a later version.

3.13. L Firmware Management

Table 357. Test Case Id: TC_L_01_CSMS

Test case name	Secure Firmware Update - Installation successful
Test case Id	TC_L_01_CSMS
Use case Id(s)	L01
Requirement(s)	L01.FR.01,L01.FR.11,L01.FR.15
System under test	CSMS
Description	The CSMS is able to request the Charging Station to securely download and install a new firmware by sending an UpdateFirmwareRequest with a signingCertificate.
Purpose	To verify if the CSMS is able to request the Charging Station to securely download and install a new firmware.
Prerequisite(s)	N/a
Before (Preparations)	Configuration State: N/a
	Memory State: N/a
	Reusable State(s): N/a

Test case name	Secure Firmware Update - Installation successful	
Main (Test scenario)	<p>Charging Station</p> <p>2. The OCTT responds with a UpdateFirmwareResponse With status Accepted</p> <p>3. The OCTT sends a FirmwareStatusNotificationRequest. With status Downloading</p> <p>5. The OCTT sends a FirmwareStatusNotificationRequest. With status Downloaded</p> <p>7. The OCTT sends a FirmwareStatusNotificationRequest. With status SignatureVerified</p> <p>9. The OCTT sends a FirmwareStatusNotificationRequest. With status Installing</p> <p>11. The OCTT sends a FirmwareStatusNotificationRequest. With status InstallRebooting</p> <p>13. The OCTT sends a BootNotificationRequest With reason FirmwareUpdate</p> <p>15. The OCTT notifies the CSMS about the current state of all connectors.</p> <p>Message: StatusNotificationRequest connectorStatus Available Message: NotifyEventRequest trigger Delta actualValue "Available" component.name "Connector" variable.name "AvailabilityState"</p> <p>17. The OCTT sends a FirmwareStatusNotificationRequest. With status Installed</p>	<p>CSMS</p> <p>1. The CSMS sends a UpdateFirmwareRequest</p> <p>4. The CSMS responds with a FirmwareStatusNotificationResponse.</p> <p>6. The CSMS responds with a FirmwareStatusNotificationResponse.</p> <p>8. The CSMS responds with a FirmwareStatusNotificationResponse.</p> <p>10. The CSMS responds with a FirmwareStatusNotificationResponse.</p> <p>12. The CSMS responds with a FirmwareStatusNotificationResponse.</p> <p>14. The CSMS responds with a BootNotificationResponse</p> <p>16. The CSMS responds accordingly.</p> <p>18. The CSMS responds with a FirmwareStatusNotificationResponse.</p>
Tool validations	<ul style="list-style-type: none"> * Step 1: Message UpdateFirmwareRequest - firmware.signingCertificate <Configured signingCertificate> - firmware.signature <Configured signature> * Step 14: Message BootNotificationResponse - status Accepted <p>Post scenario validations: N/a</p>	

Table 358. Test Case Id: TC_L_02_CSMS

Test case name	Secure Firmware Update - InstallScheduled	
Test case Id	TC_L_02_CSMS	
Use case Id(s)	L01	
Requirement(s)	L01.FR.01,L01.FR.11,L01.FR.15	
System under test	CSMS	
Description	The CSMS is able to request the Charging Station to securely download and install a new firmware by sending an UpdateFirmwareRequest with a signingCertificate .	
Purpose	To verify if the CSMS is able to request the Charging Station to securely download a new firmware and install it	
Prerequisite(s)	The CSMS configuration firmware installDateTime needs to be set to a future dateTime .	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): N/a	
Main (Test scenario)	Charging Station 2. The OCTT responds with a UpdateFirmwareResponse With status Accepted 3. The OCTT sends a FirmwareStatusNotificationRequest . With status Downloading 5. The OCTT sends a FirmwareStatusNotificationRequest . With status Downloaded 7. The OCTT sends a FirmwareStatusNotificationRequest . With status SignatureVerified 9. The OCTT sends a FirmwareStatusNotificationRequest . With status InstallScheduled Note(s): - This step will be executed after the given installDateTime from step 1 has been reached. 13. The OCTT sends a FirmwareStatusNotificationRequest . With status InstallRebooting	CSMS 1. The CSMS sends a UpdateFirmwareRequest 4. The CSMS responds with a FirmwareStatusNotificationResponse . 6. The CSMS responds with a FirmwareStatusNotificationResponse . 8. The CSMS responds with a FirmwareStatusNotificationResponse . 10. The CSMS responds with a FirmwareStatusNotificationResponse . 12. The CSMS responds with a FirmwareStatusNotificationResponse . 14. The CSMS responds with a FirmwareStatusNotificationResponse .

Test case name	Secure Firmware Update - InstallScheduled	
	<p>15. The OCTT sends a BootNotificationRequest With reason FirmwareUpdate</p>	<p>16. The CSMS responds with a BootNotificationResponse</p>
	<p>17. The OCTT notifies the CSMS about the current state of all connectors.</p> <p>Message: StatusNotificationRequest connectorStatus Available</p> <p>Message: NotifyEventRequest trigger Delta actualValue "Available" component.name "Connector" variable.name "AvailabilityState"</p>	<p>18. The CSMS responds accordingly.</p>
	<p>19. The OCTT sends a FirmwareStatusNotificationRequest. With status Installed</p>	<p>20. The CSMS responds with a FirmwareStatusNotificationResponse.</p>
Tool validations	<ul style="list-style-type: none"> * Step 1: Message UpdateFirmwareRequest - firmware.installDateTime <<i>A dateTime in the future</i>> * Step 16: Message BootNotificationResponse - status Accepted <p>Post scenario validations: N/a</p>	

Table 359. Test Case Id: TC_L_03_CSMS

Test case name	Secure Firmware Update - DownloadScheduled	
Test case Id	TC_L_03_CSMS	
Use case Id(s)	L01	
Requirement(s)	L01.FR.01,L01.FR.11,L01.FR.15	
System under test	CSMS	
Description	The CSMS is able to request the Charging Station to securely download and install a new firmware by sending an UpdateFirmwareRequest with a signingCertificate .	
Purpose	To verify if the CSMS is able to request the Charging Station to schedule securely downloading a new firmware.	
Prerequisite(s)	The CSMS configuration firmware retrieveDateTime needs to be set to a future dateTime .	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): N/a	
Main (Test scenario)	Charging Station 2. The OCTT responds with a UpdateFirmwareResponse With status Accepted 3. The OCTT sends a FirmwareStatusNotificationRequest . With status DownloadScheduled 5. The OCTT sends a FirmwareStatusNotificationRequest . With status Downloading <u>Note(s):</u> - This step will be executed after the given retrieveDateTime from step 1 has been reached.	CSMS 1. The CSMS sends a UpdateFirmwareRequest 4. The CSMS responds with a FirmwareStatusNotificationResponse .
	7. The OCTT sends a FirmwareStatusNotificationRequest . With status Downloaded	6. The CSMS responds with a FirmwareStatusNotificationResponse .
	9. The OCTT sends a FirmwareStatusNotificationRequest . With status SignatureVerified	8. The CSMS responds with a FirmwareStatusNotificationResponse .
	11. The OCTT sends a FirmwareStatusNotificationRequest . With status Installing	10. The CSMS responds with a FirmwareStatusNotificationResponse .
	13. The OCTT sends a FirmwareStatusNotificationRequest . With status InstallRebooting	12. The CSMS responds with a FirmwareStatusNotificationResponse .
		14. The CSMS responds with a FirmwareStatusNotificationResponse .

Test case name	Secure Firmware Update - DownloadScheduled	
	<p>15. The OCTT sends a BootNotificationRequest With reason FirmwareUpdate</p>	<p>16. The CSMS responds with a BootNotificationResponse</p>
	<p>17. The OCTT notifies the CSMS about the current state of all connectors.</p> <p>Message: StatusNotificationRequest connectorStatus Available</p> <p>Message: NotifyEventRequest trigger Delta actualValue "Available" component.name "Connector" variable.name "AvailabilityState"</p>	<p>18. The CSMS responds accordingly.</p>
	<p>19. The OCTT sends a FirmwareStatusNotificationRequest. With status Installed</p>	<p>20. The CSMS responds with a FirmwareStatusNotificationResponse.</p>
Tool validations	<ul style="list-style-type: none"> * Step 1: Message UpdateFirmwareRequest - firmware.retrieveDateTime <A dateTime in the future> * Step 16: Message BootNotificationResponse - status Accepted <p>Post scenario validations: N/a</p>	

Table 360. Test Case Id: TC_L_04_CSMS

Test case name	Secure Firmware Update - RevokedCertificate	
Test case Id	TC_L_04_CSMS	
Use case Id(s)	L01	
Requirement(s)	L01.FR.01	
System under test	CSMS	
Description	The CSMS is able to request the Charging Station to securely download and install a new firmware by sending an UpdateFirmwareRequest with a signingCertificate .	
Purpose	To verify if the CSMS is able to handle a Charging Station reporting the firmware signing certificate is revoked.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The OCTT responds with a UpdateFirmwareResponse With status RevokedCertificate	1. The CSMS sends a UpdateFirmwareRequest
Tool validations	N/a	
	Post scenario validations: N/a	

Table 361. Test Case Id: TC_L_05_CSMS

Test case name	Secure Firmware Update - InvalidCertificate	
Test case Id	TC_L_05_CSMS	
Use case Id(s)	L01	
Requirement(s)	L01.FR.01	
System under test	CSMS	
Description	The CSMS is able to request the Charging Station to securely download and install a new firmware by sending an UpdateFirmwareRequest with a signingCertificate .	
Purpose	To verify if the CSMS is able to handle a Charging Station reporting the firmware signing certificate is invalid.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The OCTT responds with a UpdateFirmwareResponse With status InvalidCertificate	1. The CSMS sends a UpdateFirmwareRequest
Tool validations	N/a	
	Post scenario validations: N/a	

Table 362. Test Case Id: TC_L_06_CSMS

Test case name	Secure Firmware Update - InvalidSignature	
Test case Id	TC_L_06_CSMS	
Use case Id(s)	L01	
Requirement(s)	L01.FR.01,L01.FR.11	
System under test	CSMS	
Description	The CSMS is able to request the Charging Station to securely download and install a new firmware by sending an UpdateFirmwareRequest with a signingCertificate .	
Purpose	To verify if the CSMS is able to handle a Charging Station reporting the signature is invalid.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The OCTT responds with a UpdateFirmwareResponse With status Accepted	1. The CSMS sends a UpdateFirmwareRequest
	3. The OCTT sends a FirmwareStatusNotificationRequest . With status Downloading	4. The CSMS responds with a FirmwareStatusNotificationResponse .
	5. The OCTT sends a FirmwareStatusNotificationRequest . With status Downloaded	6. The CSMS responds with a FirmwareStatusNotificationResponse .
	7. The OCTT sends a FirmwareStatusNotificationRequest . With status InvalidSignature	8. The CSMS responds with a FirmwareStatusNotificationResponse .
	N/a	
Tool validations	Post scenario validations: N/a	

Table 363. Test Case Id: TC_L_07_CSMS

Test case name	Secure Firmware Update - DownloadFailed	
Test case Id	TC_L_07_CSMS	
Use case Id(s)	L01	
Requirement(s)	L01.FR.01,L01.FR.11	
System under test	CSMS	
Description	The CSMS is able to request the Charging Station to securely download and install a new firmware by sending an UpdateFirmwareRequest with a signingCertificate .	
Purpose	To verify if the CSMS is able to handle a Charging Station reporting it failed to download the firmware.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The OCTT responds with a UpdateFirmwareResponse With status Accepted	1. The CSMS sends a UpdateFirmwareRequest
	3. The OCTT sends a FirmwareStatusNotificationRequest . With status Downloading	4. The CSMS responds with a FirmwareStatusNotificationResponse .
	5. The OCTT sends a FirmwareStatusNotificationRequest . With status DownloadFailed	6. The CSMS responds with a FirmwareStatusNotificationResponse .
Tool validations	N/a	
	Post scenario validations: N/a	

Table 364. Test Case Id: TC_L_08_CSMS

Test case name	Secure Firmware Update - InstallVerificationFailed	
Test case Id	TC_L_08_CSMS	
Use case Id(s)	L01	
Requirement(s)	L01.FR.01,L01.FR.11	
System under test	CSMS	
Description	The CSMS is able to request the Charging Station to securely download and install a new firmware by sending an UpdateFirmwareRequest with a signingCertificate .	
Purpose	To verify if the CSMS is able to handle a Charging Station reporting the verification of the firmware failed during installation.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): N/a	
Main (Test scenario)	Charging Station 2. The OCTT responds with a UpdateFirmwareResponse With status Accepted 3. The OCTT sends a FirmwareStatusNotificationRequest . With status Downloading 5. The OCTT sends a FirmwareStatusNotificationRequest . With status Downloaded 7. The OCTT sends a FirmwareStatusNotificationRequest . With status SignatureVerified 9. The OCTT sends a FirmwareStatusNotificationRequest . With status Installing 11. The OCTT sends a FirmwareStatusNotificationRequest . With status InstallVerificationFailed	CSMS 1. The CSMS sends a UpdateFirmwareRequest 4. The CSMS responds with a FirmwareStatusNotificationResponse . 6. The CSMS responds with a FirmwareStatusNotificationResponse . 8. The CSMS responds with a FirmwareStatusNotificationResponse . 10. The CSMS responds with a FirmwareStatusNotificationResponse . 12. The CSMS responds with a FirmwareStatusNotificationResponse .
Tool validations	N/a Post scenario validations: N/a	

Table 365. Test Case Id: TC_L_09_CSMS

Test case name	Secure Firmware Update - InstallationFailed	
Test case Id	TC_L_09_CSMS	
Use case Id(s)	L01	
Requirement(s)	L01.FR.01,L01.FR.11	
System under test	CSMS	
Description	The CSMS is able to request the Charging Station to securely download and install a new firmware by sending an UpdateFirmwareRequest with a signingCertificate .	
Purpose	To verify if the CSMS is able to handle a Charging Station reporting the installation of the firmware failed.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The OCTT responds with a UpdateFirmwareResponse With status Accepted	1. The CSMS sends a UpdateFirmwareRequest
	3. The OCTT sends a FirmwareStatusNotificationRequest . With status Downloading	4. The CSMS responds with a FirmwareStatusNotificationResponse .
	5. The OCTT sends a FirmwareStatusNotificationRequest . With status Downloaded	6. The CSMS responds with a FirmwareStatusNotificationResponse .
	7. The OCTT sends a FirmwareStatusNotificationRequest . With status SignatureVerified	8. The CSMS responds with a FirmwareStatusNotificationResponse .
	9. The OCTT sends a FirmwareStatusNotificationRequest . With status Installing	10. The CSMS responds with a FirmwareStatusNotificationResponse .
	11. The OCTT sends a FirmwareStatusNotificationRequest . With status InstallRebooting	12. The CSMS responds with a FirmwareStatusNotificationResponse .
	13. The OCTT sends a BootNotificationRequest With reason FirmwareUpdate	14. The CSMS responds with a BootNotificationResponse
	15. The OCTT notifies the CSMS about the current state of all connectors. Message: StatusNotificationRequest connectorStatus Available Message: NotifyEventRequest trigger Delta actualValue "Available" component.name "Connector" variable.name "AvailabilityState"	16. The CSMS responds accordingly.
	17. The OCTT sends a FirmwareStatusNotificationRequest . With status InstallationFailed	18. The CSMS responds with a FirmwareStatusNotificationResponse .

Test case name	Secure Firmware Update - InstallationFailed
Tool validations	* Step 14: Message BootNotificationResponse - status Accepted Post scenario validations: N/a

Table 366. Test Case Id: TC_L_10_CSMS

Test case name	Secure Firmware Update - AcceptedCanceled	
Test case Id	TC_L_10_CSMS	
Use case Id(s)	L01	
Requirement(s)	L01.FR.01,L01.FR.11,L01.FR.24	
System under test	CSMS	
Description	The CSMS is able to request the Charging Station to securely download and install a new firmware by sending an UpdateFirmwareRequest with a signingCertificate .	
Purpose	To verify if the CSMS is able to handle a Charging Station reporting an ongoing installation of a firmware was canceled and it is now starting the new firmware update.	
Prerequisite(s)	The CSMS is able to request a new firmware update, while there is already one ongoing on the Charging Station.	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The OCTT responds with a UpdateFirmwareResponse With status Accepted	1. The CSMS sends a UpdateFirmwareRequest
	3. The OCTT sends a FirmwareStatusNotificationRequest . With status Downloading	4. The CSMS responds with a FirmwareStatusNotificationResponse .
	6. The OCTT responds with a UpdateFirmwareResponse With status AcceptedCanceled	5. The CSMS sends a UpdateFirmwareRequest
	7. The OCTT sends a FirmwareStatusNotificationRequest . With status Downloading	8. The CSMS responds with a FirmwareStatusNotificationResponse .
	9. The OCTT sends a FirmwareStatusNotificationRequest . With status Downloaded	10. The CSMS responds with a FirmwareStatusNotificationResponse .
	11. The OCTT sends a FirmwareStatusNotificationRequest . With status SignatureVerified	12. The CSMS responds with a FirmwareStatusNotificationResponse .
	13. The OCTT sends a FirmwareStatusNotificationRequest . With status Installing	14. The CSMS responds with a FirmwareStatusNotificationResponse .
	15. The OCTT sends a FirmwareStatusNotificationRequest . With status InstallRebooting	16. The CSMS responds with a FirmwareStatusNotificationResponse .

Test case name	Secure Firmware Update - AcceptedCanceled	
	17. The OCTT sends a BootNotificationRequest With reason FirmwareUpdate	18. The CSMS responds with a BootNotificationResponse
	19. The OCTT notifies the CSMS about the current state of all connectors. Message: StatusNotificationRequest connectorStatus Available Message: NotifyEventRequest trigger Delta actualValue "Available" component.name "Connector" variable.name "AvailabilityState"	20. The CSMS responds accordingly.
	21. The OCTT sends a FirmwareStatusNotificationRequest . With status Installed	22. The CSMS responds with a FirmwareStatusNotificationResponse .
Tool validations	* Step 18: Message BootNotificationResponse - status Accepted Post scenario validations: N/a	

Table 367. Test Case Id: TC_L_11_CSMS

Test case name	Secure Firmware Update - Unable to cancel	
Test case Id	TC_L_11_CSMS	
Use case Id(s)	L01	
Requirement(s)	L01.FR.01,L01.FR.11,L01.FR.27	
System under test	CSMS	
Description	The CSMS is able to request the Charging Station to securely download and install a new firmware by sending an UpdateFirmwareRequest with a signingCertificate .	
Purpose	To verify if the CSMS is able to handle a Charging Station reporting the ongoing installation of a firmware cannot be canceled.	
Prerequisite(s)	The CSMS is able to request a new firmware update, while there is already one ongoing on the Charging Station.	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The OCTT responds with a UpdateFirmwareResponse With status Accepted	1. The CSMS sends a UpdateFirmwareRequest
	3. The OCTT sends a FirmwareStatusNotificationRequest . With status Downloading	4. The CSMS responds with a FirmwareStatusNotificationResponse .
	6. The OCTT responds with a UpdateFirmwareResponse With status Rejected	5. The CSMS sends a UpdateFirmwareRequest
	7. The OCTT sends a FirmwareStatusNotificationRequest . With status Downloaded	8. The CSMS responds with a FirmwareStatusNotificationResponse .
	9. The OCTT sends a FirmwareStatusNotificationRequest . With status SignatureVerified	10. The CSMS responds with a FirmwareStatusNotificationResponse .
	11. The OCTT sends a FirmwareStatusNotificationRequest . With status Installing	12. The CSMS responds with a FirmwareStatusNotificationResponse .
	13. The OCTT sends a FirmwareStatusNotificationRequest . With status InstallRebooting	14. The CSMS responds with a FirmwareStatusNotificationResponse .
	15. The OCTT sends a BootNotificationRequest With reason FirmwareUpdate	16. The CSMS responds with a BootNotificationResponse

Test case name	Secure Firmware Update - Unable to cancel	
	<p>17. The OCTT notifies the CSMS about the current state of all connectors.</p> <p>Message: StatusNotificationRequest connectorStatus Available</p> <p>Message: NotifyEventRequest trigger Delta actualValue "Available" component.name "Connector" variable.name "AvailabilityState"</p>	18. The CSMS responds accordingly.
	<p>19. The OCTT sends a FirmwareStatusNotificationRequest. With status Installed</p>	20. The CSMS responds with a FirmwareStatusNotificationResponse .
Tool validations	<ul style="list-style-type: none"> * Step 16: Message BootNotificationResponse - status Accepted <p>Post scenario validations: N/a</p>	

Table 368. Test Case Id: TC_L_13_CSMS

Test case name	Secure Firmware Update - Unable to download/install firmware with ongoing transaction - AllowNewSessionsPendingFirmwareUpdate is false	
Test case Id	TC_L_13_CSMS	
Use case Id(s)	L01	
Requirement(s)	L01.FR.01,L01.FR.11	
System under test	CSMS	
Description	The CSMS is able to request the Charging Station to securely download and install a new firmware by sending an UpdateFirmwareRequest with a signingCertificate .	
Purpose	To verify if the CSMS is able to handle a Charging Station setting connectors to Unavailable while preparing a firmware update when there is a transaction ongoing.	
Prerequisite(s)	The CSMS is able to request a new firmware update when there is a transaction ongoing on the Charging Station.	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): State is <i>EnergyTransferStarted</i>	
Main (Test scenario)	Charging Station 2. The OCTT responds with a UpdateFirmwareResponse With status Accepted 3. The OCTT sends a FirmwareStatusNotificationRequest . With status DownloadScheduled 5. The OCTT notifies the CSMS about the state change of all connectors that don't have a running transaction. Message: StatusNotificationRequest connectorStatus Unavailable Message: NotifyEventRequest trigger Delta actualValue "Unavailable" component.name "Connector" variable.name "AvailabilityState"	CSMS 1. The CSMS sends a UpdateFirmwareRequest 4. The CSMS responds with a FirmwareStatusNotificationResponse . 6. The CSMS responds accordingly.
	 7. Execute Reusable State StopAuthorized <u>Note(s)</u> Wait <configured transaction duration> before executing this step	
	8. Execute Reusable State EVConnectedPostSession	
	9. Execute Reusable State EVDisconnected	
	10. The OCTT sends a FirmwareStatusNotificationRequest . With status Downloading <u>Note(s)</u> : - This step will be executed after the given retrieveDateTime from step 1 has been reached.	11. The CSMS responds with a FirmwareStatusNotificationResponse .

Test case name	Secure Firmware Update - Unable to download/install firmware with ongoing transaction - AllowNewSessionsPendingFirmwareUpdate is false	
	12. The OCTT sends a FirmwareStatusNotificationRequest . With status Downloaded	13. The CSMS responds with a FirmwareStatusNotificationResponse .
	14. The OCTT sends a FirmwareStatusNotificationRequest . With status SignatureVerified	15. The CSMS responds with a FirmwareStatusNotificationResponse .
	16. The OCTT sends a FirmwareStatusNotificationRequest . With status Installing	17. The CSMS responds with a FirmwareStatusNotificationResponse .
	18. The OCTT sends a FirmwareStatusNotificationRequest . With status InstallRebooting	19. The CSMS responds with a FirmwareStatusNotificationResponse .
	20. The OCTT sends a BootNotificationRequest With reason FirmwareUpdate	21. The CSMS responds with a BootNotificationResponse
	22. The OCTT notifies the CSMS about the current state of all connectors. Message: StatusNotificationRequest connectorStatus Available Message: NotifyEventRequest trigger Delta actualValue "Available" component.name "Connector" variable.name "AvailabilityState"	23. The CSMS responds accordingly.
	24. The OCTT sends a FirmwareStatusNotificationRequest . With status Installed	25. The CSMS responds with a FirmwareStatusNotificationResponse .
Tool validations	<ul style="list-style-type: none"> * Step 1: Message UpdateFirmwareRequest - firmware.signingCertificate <configured signingCertificate> * Step 19: Message BootNotificationResponse - status Accepted <p>Post scenario validations: N/a</p>	

3.14. M ISO IEC 15118 CertificateManagement

Table 369. Test Case Id: TC_M_01_CSMS

Test case name	Install CA certificate - CSMSRootCertificate	
Test case Id	TC_M_01_CSMS	
Use case Id(s)	M05	
Requirement(s)	M05.FR.01	
System under test	CSMS	
Description	The CSMS is able to request the Charging Station to install new Root CA certificates using the InstallCertificateRequest message.	
Purpose	To verify if the CSMS is able to request a Charging Station to install a new CSMSRootCertificate.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	1. Execute Reusable State <i>CertificateInstalled</i> for certificateType CSMSRootCertificate	
Tool validations	N.a	
	Post scenario validations: N/a	

Table 370. Test Case Id: TC_M_02_CSMS

Test case name	Install CA certificate - ManufacturerRootCertificate	
Test case Id	TC_M_02_CSMS	
Use case Id(s)	M05	
Requirement(s)	M05.FR.01	
System under test	CSMS	
Description	The CSMS is able to request the Charging Station to install new Root CA certificates using the InstallCertificateRequest message.	
Purpose	To verify if the CSMS is able to request a Charging Station to install a new ManufacturerRootCertificate.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	1. Execute Reusable State <i>CertificateInstalled</i> for certificateType <i>ManufacturerRootCertificate</i>	
Tool validations	N/a	
	Post scenario validations: N/a	

Table 371. Test Case Id: TC_M_05_CSMS

Test case name	Install CA certificate - Failed	
Test case Id	TC_M_05_CSMS	
Use case Id(s)	M05	
Requirement(s)	M05.FR.01,M05.FR.03	
System under test	CSMS	
Description	The CSMS is able to request the Charging Station to install new Root CA certificates using the InstallCertificateRequest message.	
Purpose	To verify if the CSMS is able to handle a Charging Station reporting it failed to install the requested certificate.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	<u>Manual Action:</u> Trigger the CSMS to send an InstallCertificateRequest with certificateType CSMSRootCertificate .	
	2. The OCTT responds with a InstallCertificateResponse With status is <i>Failed</i>	1. The CSMS sends a InstallCertificateRequest

Table 372. Test Case Id: TC_M_13_CSMS

Test case name	Retrieve certificates from Charging Station - ManufacturerRootCertificate	
Test case Id	TC_M_13_CSMS	
Use case Id(s)	M03	
Requirement(s)	M03.FR.01	
System under test	CSMS	
Description	The CSMS is able to retrieve the certificates installed at the Charging Station using the GetInstalledCertificatesRequest message.	
Purpose	To verify if the CSMS is able to retrieve the hashData from all ManufacturerRootCertificate stored at the Charging Station.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	1. Execute Reusable State GetInstalledCertificates for certificateType ManufacturerRootCertificate	
Tool validations	N/a	
	Post scenario validations: N/a	

Table 373. Test Case Id: TC_M_18_CSMS

Test case name	Retrieve certificates from Charging Station - All certificateTypes	
Test case Id	TC_M_18_CSMS	
Use case Id(s)	M03	
Requirement(s)	M03.FR.01	
System under test	CSMS	
Description	The CSMS is able to retrieve the certificates installed at the Charging Station using the GetInstalledCertificateIdsRequest message.	
Purpose	To verify if the CSMS is able to retrieve the hashData from all Root CA and V2GCertificateChain certificates stored at the Charging Station.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	<u>Manual Action:</u> Trigger the CSMS to send a GetInstalledCertificateIdsRequest without certificateType. 2. The OCTT responds with a GetInstalledCertificateIdsResponse With status is Accepted certificateHashDataChain contains <The hashData of all certificates stored at the OCTT truststore>	
Tool validations	* Step 1: Message: GetInstalledCertificateIdsRequest - certificateType is omitted	
	Post scenario validations: N/a	

Table 374. Test Case Id: TC_M_19_CSMS

Test case name	Retrieve certificates from Charging Station - No matching certificate found	
Test case Id	TC_M_19_CSMS	
Use case Id(s)	M03	
Requirement(s)	M03.FR.01,M03.FR.02	
System under test	CSMS	
Description	The CSMS is able to retrieve the certificates installed at the Charging Station using the GetInstalledCertificateIdsRequest message.	
Purpose	To verify if the CSMS is able to handle a response from the Charging Station indicating it was not able to find a certificate for the requested criteria.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	<u>Manual Action:</u> Trigger the CSMS to send a GetInstalledCertificateIdsRequest with certificateType ManufacturerRootCertificate. 2. The OCTT responds with a GetInstalledCertificateIdsResponse With status is <i>NotFound</i> certificateHashDataChain is omitted.	
Tool validations	* Step 1: Message: GetInstalledCertificateIdsRequest - certificateType is <i>ManufacturerRootCertificate</i> Post scenario validations: N/a	

Table 375. Test Case Id: TC_M_20_CSMS

Test case name	Delete a certificate from a Charging Station - Success	
Test case Id	TC_M_20_CSMS	
Use case Id(s)	M04	
Requirement(s)	M04.FR.01,M04.FR.07	
System under test	CSMS	
Description	The CSMS is able to request the Charging Station to delete an installed certificate using the DeleteCertificateRequest message, using all available hash algorithms, including SHA256, SHA384, and SHA512.	
Purpose	To verify if CSMS is able to request a Charging Station to delete an installed certificate, using all available hash algorithms, including SHA256, SHA384, and SHA512.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): N/a	
Main (Test scenario)	Charging Station 1. CertificateInstalled with certificateType CSMSRootCertificate. <u>Manual Action:</u> Request the CSMS to send a DeleteCertificateRequest .	CSMS 2. The OCTT responds with a GetInstalledCertificateIdsRequest
	3. The OCTT responds with a GetInstalledCertificateIdsResponse With status is Accepted certificateHashDataChain contains an entry with following values: certificateHashDataChain[0].certificateType is CSMSRootCertificate certificateHashDataChain[0].certificateHashData.hashAlgorithm is SHA256	
	5. The OCTT responds with a DeleteCertificateResponse With status is Accepted	4. The CSMS sends a DeleteCertificateRequest
	<u>Note(s):</u> - Steps 1 - 5 will be repeated for each hash algorithm (SHA256, SHA384, SHA512).	
Tool validations	* Step 2: Message: GetInstalledCertificateIdsRequest - certificateType contains CSMSRootCertificate OR is omitted. * Step 4: Message: DeleteCertificateRequest - certificateHashData is <Returned certificateHashData at Step 3>. Post scenario validations: N/a	

Table 376. Test Case Id: TC_M_21_CSMS

Test case name	Delete a certificate from a Charging Station - Failed	
Test case Id	TC_M_21_CSMS	
Use case Id(s)	M04	
Requirement(s)	M04.FR.01,M04.FR.07	
System under test	CSMS	
Description	The CSMS is able to request the Charging Station to delete an installed certificate using the DeleteCertificateRequest message.	
Purpose	To verify if CSMS is able to handle a Charging Station that fails to delete an installed certificate.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): <i>CertificateInstalled</i> with certificateType CSMSRootCertificate.	
Main (Test scenario)	Charging Station	CSMS
	<u>Manual Action:</u> Request the CSMS to send a DeleteCertificateRequest.	
	2. The OCTT responds with a GetInstalledCertificateIdsResponse With status is Accepted certificateHashDataChain contains an entry with following values: certificateHashDataChain[0].certificateType is CSMSRootCertificate certificateHashDataChain[0].certificateHashData.ha shAlgorithm is SHA256	1. The CSMS sends a GetInstalledCertificateIdsRequest
	4. The OCTT responds with a DeleteCertificateResponse With status is Failed	3. The CSMS sends a DeleteCertificateRequest
Tool validations	<ul style="list-style-type: none"> * Step 1: Message: GetInstalledCertificateIdsRequest - certificateType contains CSMSRootCertificate OR is omitted. * Step 3: Message: DeleteCertificateRequest - certificateHashData contains <Returned certificateHashData at Step 2>. <p>Post scenario validations: N/a</p>	

3.15. N Diagnostics

Table 377. Test Case Id: TC_N_25_CSMS

Test case name	Retrieve Log Information - Diagnostics Log - Success	
Test case Id	TC_N_25_CSMS	
Use case Id(s)	N01	
Requirement(s)	N/a	
System under test	CSMS	
Description	This test case covers the functionality of getting log information from a Charging Station. The CSMS can request a Charging Station to upload a file with log information to a given location (URL). The format of this log file is not prescribed. The Charging Station successfully uploads a log file and gives information about the status of the upload by sending status notifications to the CSMS.	
Purpose	To verify if the CSMS is able to request a charging station to successfully upload a log as described at the OCPP specification.	
Prerequisite(s)	Charging Station has log information available.	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The OCTT responds with a GetLogResponse with status Accepted	1. The CSMS sends a GetLogRequest
	3. The OCTT sends a LogStatusNotificationRequest with - status Uploading - requestId Same Id as the GetLogRequest	4. The CSMS responds with a LogStatusNotificationResponse .
	5. The OCTT sends a LogStatusNotificationRequest with - status Uploaded - requestId Same Id as the GetLogRequest	6. The CSMS responds with a LogStatusNotificationResponse .
	Tool validations	* Step 1: Message GetLogRequest - logType DiagnosticsLog Post scenario validations: - N/a

Table 378. Test Case Id: TC_N_27_CSMS

Test case name	Get Customer Information - Accepted + data	
Test case Id	TC_N_27_CSMS	
Use case Id(s)	N09	
Requirement(s)	N09.FR.01, N09.FR.04	
System under test	CSMS	
Description	The CSMS sends a message to the Charging Station to retrieve IdToken customer information, for example to be compliant with local privacy laws. The Charging Station notifies the CSMS by sending one or more reports.	
Purpose	To verify if the CSMS sends the request correctly and responds on the notifies as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The OCTT responds with a CustomerInformationResponse with status Accepted	1. The CSMS sends a CustomerInformationRequest
	3. The OCTT sends a NotifyCustomerInformationRequest	4. The CSMS responds with a NotifyCustomerInformationResponse .
Tool validations	<ul style="list-style-type: none"> * Step 1: Message CustomerInformationRequest - report true - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type> 	
	Post scenario validations: <ul style="list-style-type: none"> - N/a 	

Table 379. Test Case Id: TC_N_28_CSMS

Test case name	Get Customer Information - Accepted + no data	
Test case Id	TC_N_28_CSMS	
Use case Id(s)	N09	
Requirement(s)	N09.FR.01, N09.FR.04	
System under test	CSMS	
Description	The CSMS sends a message to the Charging Station to retrieve IdToken customer information, for example to be compliant with local privacy laws. The Charging Station notifies the CSMS by sending one or more reports.	
Purpose	To verify if the CSMS sends the request correctly and responds on the notifies as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The OCTT responds with a CustomerInformationResponse with status Accepted	1. The CSMS sends a CustomerInformationRequest
	3. The OCTT sends a NotifyCustomerInformationRequest	4. The CSMS responds with a NotifyCustomerInformationResponse .
Tool validations	<ul style="list-style-type: none"> * Step 1: Message CustomerInformationRequest - report true - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type> 	
	Post scenario validations: <ul style="list-style-type: none"> - N/a 	

Table 380. Test Case Id: TC_N_29_CSMS

Test case name	Get Customer Information - Not Accepted	
Test case Id	TC_N_29_CSMS	
Use case Id(s)	N09	
Requirement(s)	N09.FR.01, N09.FR.04	
System under test	CSMS	
Description	The CSMS sends a message to the Charging Station to retrieve IdToken customer information, but the Charging Station rejects the request.	
Purpose	To verify if the CSMS sends the request correctly as described at the OCPP specification, and can handle the Charging Station rejecting the request.	
Prerequisite(s)	n/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The OCTT responds with a CustomerInformationResponse with status Rejected	1. The CSMS sends a CustomerInformationRequest
Tool validations	<ul style="list-style-type: none"> * Step 1: <p>Message CustomerInformationRequest</p> <ul style="list-style-type: none"> - report true - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type> 	
	<p>Post scenario validations:</p> <ul style="list-style-type: none"> - N/a 	

Table 381. Test Case Id: TC_N_30_CSMS

Test case name	Clear Customer Information - Clear and report + data	
Test case Id	TC_N_30_CSMS	
Use case Id(s)	N10	
Requirement(s)	N10.FR.08	
System under test	CSMS	
Description	The CSMS sends a message to the Charging Station to clear (and retrieve) IdToken customer information, for example to be compliant with local privacy laws. The Charging Station notifies the CSMS by sending one or more reports.	
Purpose	To verify if the CSMS sends the request correctly and responds on the notifies as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	Configuration State: N/a Memory State: N/a Charging State: N/a	
Main (Test scenario)	Charging Station 2. The OCTT responds with a CustomerInformationResponse with status Accepted 3. The OCTT sends a NotifyCustomerInformationRequest	CSMS 1. The CSMS sends a CustomerInformationRequest 4. The CSMS responds with a NotifyCustomerInformationResponse .
Tool validations	* Step 1: Message CustomerInformationRequest - report true - clear true - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type> Post scenario validations: - N/a	

Table 382. Test Case Id: TC_N_31_CSMS

Test case name	Clear Customer Information - Clear and report + no data	
Test case Id	TC_N_31_CSMS	
Use case Id(s)	N10	
Requirement(s)	N10.FR.08	
System under test	CSMS	
Description	The CSMS sends a message to the Charging Station to clear (and retrieve) IdToken customer information, for example to be compliant with local privacy laws. The Charging Station notifies the CSMS by sending one or more reports.	
Purpose	To verify if the CSMS sends the request correctly and responds on the notifies as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The OCTT responds with a CustomerInformationResponse with status Accepted	1. The CSMS sends a CustomerInformationRequest
	3. The OCTT sends a NotifyCustomerInformationRequest	4. The CSMS responds with a NotifyCustomerInformationResponse .
Tool validations	<p>* Step 1: Message CustomerInformationRequest - report true - clear true - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type></p> <p>Post scenario validations: - N/a</p>	

Table 383. Test Case Id: TC_N_32_CSMS

Test case name	Clear Customer Information - Clear and no report	
Test case Id	TC_N_32_CSMS	
Use case Id(s)	N10	
Requirement(s)	N10.FR.08	
System under test	CSMS	
Description	The CSMS sends a message to the Charging Station to clear IdToken customer information, for example to be compliant with local privacy laws.	
Purpose	To verify if the CSMS sends the request correctly.	
Prerequisite(s)	n/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The OCTT responds with a CustomerInformationResponse with status Accepted	1. The CSMS sends a CustomerInformationRequest
Tool validations	<ul style="list-style-type: none"> * Step 1: Message CustomerInformationRequest - report false - clear true - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type> 	
	Post scenario validations: <ul style="list-style-type: none"> - N/a 	

Table 384. Test Case Id: TC_N_62_CSMS

Test case name	Clear Customer Information - Clear and report - customerIdentifier	
Test case Id	TC_N_62_CSMS	
Use case Id(s)	N10	
Requirement(s)	N10.FR.08	
System under test	CSMS	
Description	The CSMS sends a message to the Charging Station to clear (and retrieve) raw customer information, for example to be compliant with local privacy laws. The Charging Station notifies the CSMS by sending one or more reports.	
Purpose	To verify if the CSMS sends the request correctly and responds on the notifies as described at the OCPP specification.	
Prerequisite(s)	The CSMS supports retrieving / deleting CustomerInformation - CustomerIdentifier	
Before (Preparations)	Configuration State: N/a Memory State: N/a Charging State: N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The OCTT responds with a CustomerInformationResponse with status Accepted	1. The CSMS sends a CustomerInformationRequest
	3. The OCTT sends a NotifyCustomerInformationRequest	4. The CSMS responds with a NotifyCustomerInformationResponse
Tool validations	* Step 1: Message CustomerInformationRequest - report true - clear true - customerIdentifier "OpenChargeAlliance" Post scenario validations: - N/a	

Table 385. Test Case Id: TC_N_34_CSMS

Test case name	Retrieve Log Information - Rejected	
Test case Id	TC_N_34_CSMS	
Use case Id(s)	N01	
Requirement(s)	N/a	
System under test	CSMS	
Description	This test case covers the functionality of getting log information from a Charging Station. The CSMS can request a Charging Station to upload a file with log information to a given location (URL). The format of this log file is not prescribed. The Charging Station successfully uploads a log file and gives information about the status of the upload by sending status notifications to the CSMS.	
Purpose	To verify if the CSMS is able to request a charging station to successfully upload a log as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The OCTT responds with a GetLogResponse with status Rejected	1. The CSMS sends a GetLogRequest
Tool validations	N/a	
	Post scenario validations: - N/a	

Table 386. Test Case Id: TC_N_35_CSMS

Test case name	Retrieve Log Information - Security Log - Success	
Test case Id	TC_N_35_CSMS	
Use case Id(s)	N01	
Requirement(s)		
System under test	CSMS	
Description	This test case covers the functionality of getting log information from a Charging Station. The CSMS can request a Charging Station to upload a file with log information to a given location (URL). The format of this log file is not prescribed. The Charging Station successfully uploads a log file and gives information about the status of the upload by sending status notifications to the CSMS.	
Purpose	To verify if the CSMS is able to request a charging station to successfully upload a log as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: Charging Station has log information available.	
	Charging State: N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The OCTT responds with a GetLogResponse with status Accepted	1. The CSMS sends a GetLogRequest
	3. The OCTT sends a LogStatusNotificationRequest with - status Uploading - requestId Same Id as the GetLogRequest	4. The CSMS responds with a LogStatusNotificationResponse .
	5. The OCTT sends a LogStatusNotificationRequest with - status Uploaded - requestId Same Id as the GetLogRequest	6. The OCTT responds with a LogStatusNotificationResponse .
	Tool validations	
<ul style="list-style-type: none"> * Step 1: Message GetLogRequest - logType SecurityLog <p>Post scenario validations: - N/a</p>		

Table 387. Test Case Id: TC_N_36_CSMS

Test case name	Retrieve Log Information - Second Request	
Test case Id	TC_N_36_CSMS	
Use case Id(s)	N01	
Requirement(s)	N/a	
System under test	CSMS	
Description	This test case covers the functionality of getting log information from a Charging Station. The CSMS can request a Charging Station to upload a file with log information to a given location (URL). The format of this log file is not prescribed. The Charging Station successfully uploads a log file and gives information about the status of the upload by sending status notifications to the CSMS.	
Purpose	To verify if the CSMS is able to request a second request while the charging station is uploading a log as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: Charging Station has log information available.	
	Charging State: N/a	
Main (Test scenario)	Charging Station	CSMS
	2. The OCTT responds with a GetLogResponse with status Accepted	1. The CSMS sends a GetLogRequest
	3. The OCTT sends a LogStatusNotificationRequest with - status Uploading - requestId Same Id as the GetLogRequest from Step 1	4. The CSMS responds with a LogStatusNotificationResponse .
	6. The OCTT responds with a GetLogResponse with status AcceptedCanceled	5. The CSMS sends a GetLogRequest
	7. The OCTT sends a LogStatusNotificationRequest with - status AcceptedCanceled - requestId Same Id as the GetLogRequest from Step 1	8. The CSMS responds with a LogStatusNotificationResponse .
	9. The OCTT sends a LogStatusNotificationRequest with - status Uploading - requestId Same Id as the GetLogRequest from Step 5	10. The CSMS responds with a LogStatusNotificationResponse .
	11. The OCTT sends a LogStatusNotificationRequest with - status Uploaded - requestId Same Id as the GetLogRequest from Step 5	12. The CSMS responds with a LogStatusNotificationResponse .
	N/a	
	Post scenario validations: - N/a	

3.16. O Display Message

This section is intentionally blank, this will be added in a later version.

3.17. P DataTransfer

Table 388. Test Case Id: TC_P_02_CSMS

Test case name	Data Transfer to the CSMS - Rejected / Unknown VendorId / Unknown MessageId	
Test case Id	TC_P_02_CSMS	
Use case Id(s)	P02	
Requirement(s)	P02.FR.06, P02.FR.07	
System under test	CSMS	
Description	The DataTransfer message to send information for functions that are not supported by OCPP.	
Purpose	To verify whether the CSMS is able to handle receiving a DataTransferRequest, even if it does not support any vendor-specific implementations.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	1. The OCTT sends a DataTransferRequest with vendorId <Configured vendorId> messageId <Configured messageId>	2. The CSMS responds with a DataTransferResponse
Tool validations	<ul style="list-style-type: none"> * Step 2: Message: DataTransferResponse - status must be <i>UnknownVendorId</i> OR <i>UnknownMessageId</i> OR <i>Rejected</i> (<i>Rejected</i> will also be allowed, because there are implementers that like to just reject the message when the Charging Station does not support any vendor-specific features.) 	
	Post scenario validations: N/a	

Table 389. Test Case Id: TC_P_03_CSMS

Test case name	CustomData - Receive custom data	
Test case Id	TC_P_03_CSMS	
Use case Id(s)	N/a	
Requirement(s)	N/a	
System under test	CSMS	
Description	Checks if the CSMS is able to receive custom data.	
Purpose	To verify whether the CSMS is able to handle receiving custom data.	
Prerequisite(s)	N/a	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	Charging Station	CSMS
	1. The OCTT sends a StatusNotificationRequest with customData <customData>	2. The CSMS responds with a StatusNotificationResponse
	3. The OCTT sends a TransactionEventRequest with customData customData transactionInfo.customData <customData>	4. The CSMS responds with a TransactionEventResponse
Tool validations	N/a	
	Post scenario validations: N/a	

3.18. Reusable states

Testcases can refer to a reusable state at the before or main stage. The steps described at the reusable state will be executed and then it will return to the testcase that called the reusable state.

Table 390. Reusable State: Booted

State	Booted	
System under test	CSMS	
Description	This state will simulate that the Charging Station is completely power cycled. The OCTT ends in a state where it is "booted" back up and is in idle mode.	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): N/a	
Main (Scenario)	Charging Station 1. The OCTT sends a BootNotificationRequest with reason PowerUp chargingStation.model <Configured model> chargingStation.vendorName <Configured vendorName>	CSMS 2. The CSMS responds with a BootNotificationResponse
	 3. The OCTT notifies the CSMS about the current state of all connectors. Message: StatusNotificationRequest with connectorStatus Available Message: NotifyEventRequest with trigger Delta actualValue "Available" component.name "Connector" variable.name "AvailabilityState"	 4. The CSMS responds accordingly.
Tool validations	* Step 2: Message: BootNotificationResponse - status Accepted	
Post condition	State is Booted	

Table 391. Reusable State: Reserved

State	Reserved	
System under test	CSMS	
Description	This state will simulate a reservation for a specified evse.	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): N/a	
Main (Scenario)	Charging Station CSMS <u>Manual Action:</u> Trigger the CSMS to send a ReserveNowRequest for specific EVSE. 2. The OCTT responds with a ReserveNowResponse With status Accepted 3. The OCTT notifies the CSMS about the current state of the connector(s) of the Specified EVSE Message: StatusNotificationRequest with connectorStatus Reserved Message: NotifyEventRequest with trigger Delta actualValue "Reserved" component.name "Connector" variable.name "AvailabilityState"	1. The CSMS sends a ReserveNowRequest 4. The CSMS responds accordingly.
Tool validations	* Step 1: Message: ReserveNowRequest - evselid must be <Specified evselid> - connectorType must be omitted - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type>	
Post condition	State is Reserved	

Table 392. Reusable State: Unavailable

State	Unavailable	
System under test	CSMS	
Description	This state will simulate that Charging Station / EVSEs / connectors are set to AvailabilityState Unavailable.	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): N/a	
Main (Scenario)	Charging Station <u>Manual Action:</u> Request the CSMS to change the availability of the specified components to Inoperative. 2. The OCTT responds with a ChangeAvailabilityResponse with status Accepted 3. The OCTT notifies the CSMS about the current state of all connectors belonging to the specified EVSE (and optionally also from the EVSE itself). Message: StatusNotificationRequest - connectorStatus <i>Unavailable</i> Message: NotifyEventRequest - trigger <i>Delta</i> - actualValue "Unavailable" - component.name "ChargingStation" / EVSE / Connector - variable.name "AvailabilityState"	CSMS 1. The CSMS sends a ChangeAvailabilityRequest 4. The CSMS responds accordingly.
Tool validations	* Step 1: Message ChangeAvailabilityRequest - operationalStatus <i>Inoperative</i> - evse <Specified evselId> - connectorId omitted	
Post condition	State is Unavailable	

Table 393. Reusable State: EVConnectedPreSession

State	EVConnectedPreSession	
System under test	CSMS	
Description	This state will simulate that the EV and EVSE of the simulated Charging Station are connected.	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): N/a	
Main (Scenario)	Charging Station 1. The OCTT notifies the CSMS about the status change of the connector Message: StatusNotificationRequest - connectorStatus is <i>Occupied</i> Message: NotifyEventRequest - trigger is <i>Delta</i> - actualValue is <i>Occupied</i> - component.name is <i>Connector</i> - variable.name is <i>AvailabilityState</i>	CSMS 2. The CSMS responds accordingly.
	3. The OCTT sends a TransactionEventRequest With triggerReason is <i>CablePluggedIn</i> transactionInfo.chargingState is <i>EVConnected</i> evse.id < <i>Configured evseld</i> > evse.connectorId < <i>Configured connectorId</i> > If State is <i>Authorized</i> then eventType is <i>Updated</i> else eventType is <i>Started</i>	4. The CSMS responds with a TransactionEventResponse
Tool validations	N/a	
Post condition	State is <i>EVConnectedPreSession</i>	

Table 394. Reusable State: Authorized

State	Authorized	
System under test	CSMS	
Description	This state will simulate that the EV Driver is locally authorizing to start a transaction on the simulated Charging Station.	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): N/a	
Main (Scenario)	Charging Station 1. The OCTT sends an AuthorizeRequest With idToken.idToken <Configured valid_idtoken_idtoken> idToken.type <Configured valid_idtoken_type> 3. The OCTT sends a TransactionEventRequest With triggerReason is Authorized idToken.idToken <Configured valid_idtoken_idtoken> idToken.type <Configured valid_idtoken_type> If State is EVConnectedPreSession then eventType is Updated else eventType is Started	CSMS 2. The CSMS responds with an AuthorizeResponse 4. The CSMS responds with a TransactionEventResponse
Tool validations	<ul style="list-style-type: none"> * Step 2: Message: AuthorizeResponse - idTokenInfo.status must be Accepted * Step 4: Message: TransactionEventResponse - idTokenInfo.status must be Accepted 	
Post condition	State is Authorized	

Table 395. Reusable State: EnergyTransferStarted

State	EnergyTransferStarted	
System under test	CSMS	
Description	This state will simulate that there is transferring energy between the EV and EVSE of the simulated Charging Station.	
Before (Preparations)	<p>Configuration State: N/a</p> <p>Memory State: N/a</p> <p>Reusable State(s): If State is NOT <i>Authorized</i> then execute Reusable State <i>Authorized</i> If EVConnected is <i>true</i>, then proceed to part 2 Else proceed to part 1.</p>	
Main (Part 1) (Scenario)	Charging Station 1. The OCTT notifies the CSMS about the status change of the connector. Message: StatusNotificationRequest - connectorStatus is <i>Occupied</i> Message: NotifyEventRequest - trigger is <i>Delta</i> - actualValue is <i>Occupied</i> - component.name is <i>Connector</i> - variable.name is <i>AvailabilityState</i>	CSMS 2. The CSMS responds accordingly.
	3. The OCTT sends a TransactionEventRequest With triggerReason is <i>CablePluggedIn</i> transactionInfo.chargingState is <i>EVConnected</i> <i>evse.id <Configured evsId></i> <i>evse.connectorId <Configured connectorId></i> eventType is <i>Updated</i>	4. The CSMS responds with a TransactionEventResponse
Tool validations	N/a	
Main (Part 2) (Scenario)	Charging Station 5. The OCTT sends a TransactionEventRequest With triggerReason is <i>ChargingStateChanged</i> transactionInfo.chargingState is <i>Charging</i> eventType is <i>Updated</i>	CSMS 6. The CSMS responds with a TransactionEventResponse
Tool validations	N/a	
Post condition	State is <i>EnergyTransferStarted</i> EVConnected is <i>true</i>	

Table 396. Reusable State: EnergyTransferSuspended

State	EnergyTransferSuspended		
System under test	CSMS		
Description	This state will simulate that the Charging Station is in a state where the energy transfer is suspended by the EV.		
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): If State is NOT <i>EnergyTransferStarted</i> then execute Reusable State <i>EnergyTransferStarted</i>		
Main (Scenario)	Charging Station <u>Notes(s):</u> <i>The tool will wait for <Configured Transaction Duration> seconds</i>	CSMS 1. The OCTT sends a TransactionEventRequest With triggerReason is <i>ChargingStateChanged</i> transactionInfo.chargingState is <i>SuspendedEV</i>	2. The CSMS responds with a TransactionEventResponse
Tool validations	N/a		
Post condition	State is <i>EnergyTransferSuspended</i>		

Table 397. Reusable State: StopAuthorized

State	StopAuthorized	
System under test	CSMS	
Description	This state will simulate that the Charging Station is in a state where the charging session is authorized to stop.	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): If State is NOT <i>EnergyTransferStarted</i> then execute Reusable State <i>EnergyTransferStarted</i>	
Main (Scenario)	Charging Station <u>Notes(s):</u> <i>The tool will wait for <Configured Transaction Duration> seconds</i>	CSMS
	1. The OCTT sends a TransactionEventRequest With triggerReason is <i>StopAuthorized</i> eventType is <i>Updated</i>	2. The CSMS responds with a TransactionEventResponse
Tool validations	* Step 2: Message: TransactionEventResponse - idTokenInfo.status must be Accepted	
Post condition	State is <i>StopAuthorized</i>	

Table 398. Reusable State: EVConnectedPostSession

State	EVConnectedPostSession	
System under test	CSMS	
Description	This state will simulate that the Charging Station is in a state where the energy transfer has been stopped and the transaction is NOT authorized to resume energy transfer without re-authorization.	
Before (Preparations)	Configuration State: N/a	
	Memory State: N/a	
	Reusable State(s): If State is NOT <i>StopAuthorized</i> then execute Reusable State StopAuthorized	
Main (Scenario)	Charging Station 1. The OCTT sends a TransactionEventRequest With triggerReason is <i>ChargingStateChanged</i> transactionInfo.chargingState is <i>EVConnected</i> eventType is <i>Updated</i>	CSMS 2. The CSMS responds with a TransactionEventResponse
Tool validations	N/a	
Post condition	State is <i>EVConnectedPostSession</i>	

Table 399. Reusable State: EVDisconnected

State	EVDisconnected	
System under test	CSMS	
Description	This state will simulate that the EV and EVSE of the simulated Charging Station are disconnected, after the charging session is authorized to stop.	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): If State is NOT <i>EVConnectedPostSession</i> then execute Reusable State <i>EVConnectedPostSession</i>	
Main (Scenario)	Charging Station 1. The OCTT notifies the CSMS about the status change of the connector. Message: StatusNotificationRequest - connectorStatus is <i>Available</i> Message: NotifyEventRequest - trigger is <i>Delta</i> - actualValue is <i>Available</i> - component.name is <i>Connector</i> - variable.name is <i>AvailabilityState</i>	CSMS 2. The CSMS responds accordingly.
	3. The OCTT sends a TransactionEventRequest With triggerReason is <i>EVCommunicationLost</i> transactionInfo.chargingState is <i>Idle</i> transactionInfo.stoppedReason is <i>EVDisconnected</i> eventType is <i>Ended</i>	4. The CSMS responds with a TransactionEventResponse
Tool validations	N/a	
Post condition	State is <i>EVDisconnected</i>	

Table 400. Reusable State: GetInstalledCertificates

State	GetInstalledCertificates	
System under test	CSMS	
Description	The hashData from installed certificates of the specified type will be retrieved from the Charging Station	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): N/a	
Main (Scenario)	Charging Station <u>Manual Action:</u> Trigger the CSMS to send a GetInstalledCertificateIdsRequest with certificateType <Specified certificateType> 2. The OCTT responds with a GetInstalledCertificateIdsResponse With status is Accepted certificateHashDataChain contains an entry with following values: certificateHashDataChain[0].certificateType is <Specified certificateType> certificateHashDataChain[0].certificateHashData contains <i><HashData from the configured certificate of the specified certificateType></i>	CSMS 1. The CSMS sends a GetInstalledCertificateIdsRequest
Tool validations	* Step 1: Message: GetInstalledCertificateIdsRequest - certificateType must be <Specified certificateType>	
Post condition	Certificate of the specified certificateType is retrieved from the Charging Station.	

Table 401. Reusable State: CertificateInstalled

State	CertificateInstalled	
System under test	CSMS	
Description	A pre configured certificate of the specified certificateType will be installed.	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): N/a	
Main (Scenario)	Charging Station Manual Action: Trigger the CSMS to send an InstallCertificateRequest with certificateType <Specified certificateType> 2. The OCTT responds with a InstallCertificateResponse With status is Accepted	CSMS 1. The CSMS sends a InstallCertificateRequest
Tool validations	* Step 1: Message: InstallCertificateRequest - certificateType must be <Specified certificateType> - certificate must be <The configured certificate of the specified certificateType.>	
Post condition	Certificate of the specified certificateType is stored at the Charging Station.	

Table 402. Reusable State: ISO15118SmartCharging

State	ISO15118SmartCharging	
System under test	CSMS	
Description		
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): N/a	
Main (Scenario)	Charging Station 1. The OCTT sends a NotifyEVChargingNeedsRequest with evseld <Configured evseld> maxScheduleTuples & chargingNeeds <Configured values from mock EV>+	CSMS 2. The CSMS responds with a NotifyEVChargingNeedsResponse .
	4. The OCTT responds with a SetChargingProfileResponse with: status Accepted	3. The CSMS sends a SetChargingProfileRequest <u>Note(s):</u> - If NotifyEVChargingNeedsResponseStatus was Processing, the OCTT will wait 60 seconds for the request
	5. The OCTT sends a NotifyEVChargingScheduleRequest with evseld <Configured evseld> chargingSchedule <ChargingSchedule provided at step 3>	6. The CSMS responds with a NotifyEVChargingScheduleResponse .
	7. The OCTT sends a TransactionEventRequest with triggerReason <ChargingStateChanged> transactionInfo.chargingState <Charging>	8. The CSMS responds with a TransactionEventResponse .

State	ISO15118SmartCharging
Tool validations	<ul style="list-style-type: none"> * Step 1: Message: NotifyEVChargingNeedsResponse - Status Accepted or Processing * Step 3: Message: SetChargingProfileRequest - chargingProfilePurpose <<i>TxProfile</i>> - transactionId <Provided transactionId from before> * Step 4: Message: NotifyEVChargingScheduleResponse - status <Accepted>
Post condition	N/a

Table 403. Memory State: RenewChargingStationCertificate

State	RenewChargingStationCertificate	
System under test	CSMS	
Description	The ChargingStationCertificate is renewed using A02/A03	
Before (Preparations)	Configuration State: N/a Memory State: N/a Reusable State(s): N/a	
Main (Test scenario)	Charging Station <u>Manual Action:</u> Request the CSMS to send a Trigger Message Request with requestedMessage <i>SignChargingStationCertificate</i>	CSMS
	2. The OCTT sends a TriggerMessageResponse with status Accepted	1. The CSMS sends a TriggerMessageRequest With requestedMessage <i>SignChargingStationCertificate</i>
	3 The OCTT sends a SignCertificateRequest	4. The CSMS responds with a SignCertificateResponse with status Accepted
	6. The OCTT sends a CertificateSignedResponse with status Accepted	5. The CSMS sends a CertificateSignedRequest With certificateChain <Certificate generated from the received CSR from step 3 and signed by the configured CSMS Root certificate> certificateType <i>ChargingStationCertificate</i>
Tool validations	* Step 1: Message: TriggerMessageRequest - requestedMessage must be <i>SignChargingStationCertificate</i> * Step 4: Message: SignCertificateResponse - status must be <i>Accepted</i> * Step 5: Message: CertificateSignedRequest - certificateChain <Certificate generated from the received CSR from step 3 and signed by the configured CSMS Root certificate> - certificateType must be <i>ChargingStationCertificate</i> Post scenario validations: N/a	