OCPP 2.1

Part 2 - Specification

Edition 1 , 2025-01-23

Table of Contents

Disclaimer

Generic

Version History

Version	**Date**	**Description**
2.1 Edition 1	2025-01-23	OCPP 2.1 Edition 1

Chapter 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 2.1 is limited to JSON. The specifications for the JSON implementation are in "Part 4 - JSON over WebSockets implementation guide".

1.1. OCPP 2.1 Edition 1

OCPP 2.1 extends OCPP 2.0.1 with additional functionality. Care has been taken to make sure that the handling of OCPP 2.1 messages continues to work exactly the same as in OCPP 2.0.1 as long as none of the added (optional) fields in those messages are used.

Chapter 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, sequence diagrams, examples 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.
AnyType	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: *the V2G messaging phase for controlling the charging process by ISO 15118*.
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.
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].

Terminology	Description
Cryptography	The ENISA Algorithms, Key Sizes and Parameters Report (OCPP Security [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	EVSE stands for "EV Supply Equipment". An EVSE is considered to be 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	Bidirectional 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, *Idle* state 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.
OCPP 2.1 DER terminology	
Cease to Energize	Halting the delivery of active power during stable and unstable conditions while restricting the exchange of reactive power.
DER Curve	A set of points that outline a curve representing the adjustments required for the Y-axis value based on the X-axis value. An example of this is the Volt-Watt curve or P(U), which determines active power based on voltage.
Distributed Energy Resource (DER)	An electricity source that is not directly linked to a bulk power system. DER comprises generators and energy storage technologies that can provide active power to an electrical power system. For the purposes of this document, a bidirectional charging station and bidirectional EV make up a DER.
Frequency Droop	A type of frequency-watt curve defined by a dead-band between under- and over-frequency, and a frequency change per unit power output change (frequency droop) for under- and over-frequencies.
Momentary Cessation	A temporary halt to active power delivery in response to voltage or system frequency disruptions, with the ability to immediately restore operation output when the voltages and frequency return to specified ranges.
Nameplate Ratings	Capabilities of a DER for sustained operation, e.g. nominal voltage (V), current (A), maximum active power (kW), apparent power (kVA), and reactive power (kvar).
Ride-Through Curve	A specific type of DER curve represented as a piecewise linear curve that defines regions associated with voltage and frequency behavior for trip, momentary cessation, and may trip.
Ride-Through	The ability to withstand voltage or frequency disruptions within specified limits and continue operating as specified.
Trip Curve	Similar to a ride-through curve, but without momentary cessation.
Trip	The inhibition of immediate service restoration, which may require disconnection.

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
DER	Distributed Energy Resource
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].

Abbreviation	Description
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 URIs 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").
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: for example a DC EVSE with a CCS2 and a CHAdeMO connector).
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	An electric vehicle (EV) is a type of vehicle that is powered by one or more electric motors, using energy stored in rechargeable batteries.
Local Controller	Device	A logical entity logically placed 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/or external systems.
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
[EN50549-1]	EN 50549-1:2019 Requirements for generating plants to be connected in parallel with distribution networks - Part 1: Connection to a LV distribution network - Generating plants up to and including Type B
[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/29264
[ISO15118-2]	Road vehicles - Vehicle to grid communication interface - Part 2: Network and application protocol requirements, First edition, 2014-04-01. https://webstore.iec.ch/publication/9273
[ISO15118-20]	Road vehicles – Vehicle to grid communication interface – Part 20: Network and application protocol requirements. https://webstore.iec.ch/publication/26347
[ISO4217]	"ISO 4217: Currency codes" http://www.iso.org/iso/home/standards/currency_codes.htm
[OCPP2.1-PART1]	"OCPP 2.1: Part 1 - Architecture & Topology". http://www.openchargealliance.org/downloads/
[OCPP2.1-PART2-APPENDIX]	"OCPP 2.1: Part 2 - Appendices". http://www.openchargealliance.org/downloads/
[OCPP2.1-PART4]	"OCPP 2.1: 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
[IEC 61850-7-420]	Communication networks and systems for power utility automation - Part 7-420: Basic communication structure - Distributed energy resources logical nodes (IEC 61850-7-420:2009,IDT)
[IEEE 1547-2018]	Standard for Interconnection and Interoperability of Distributed Energy Resources with Associated Electric Power Systems Interfaces
[IEEE 2030.5-2018]	Standard for Smart Energy Profile Application Protocol
[CSIP IEEE 2030.5]	Sunspec Alliance, Common Smart Inverter Profile, Implementation Guide for Smart Inverters, 2018, v2.1
[SAE J3072]	SAE J3072-2021 Interconnection Requirements for Onboard, Grid Support Inverter Systems

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

Reference	Description
[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
[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.

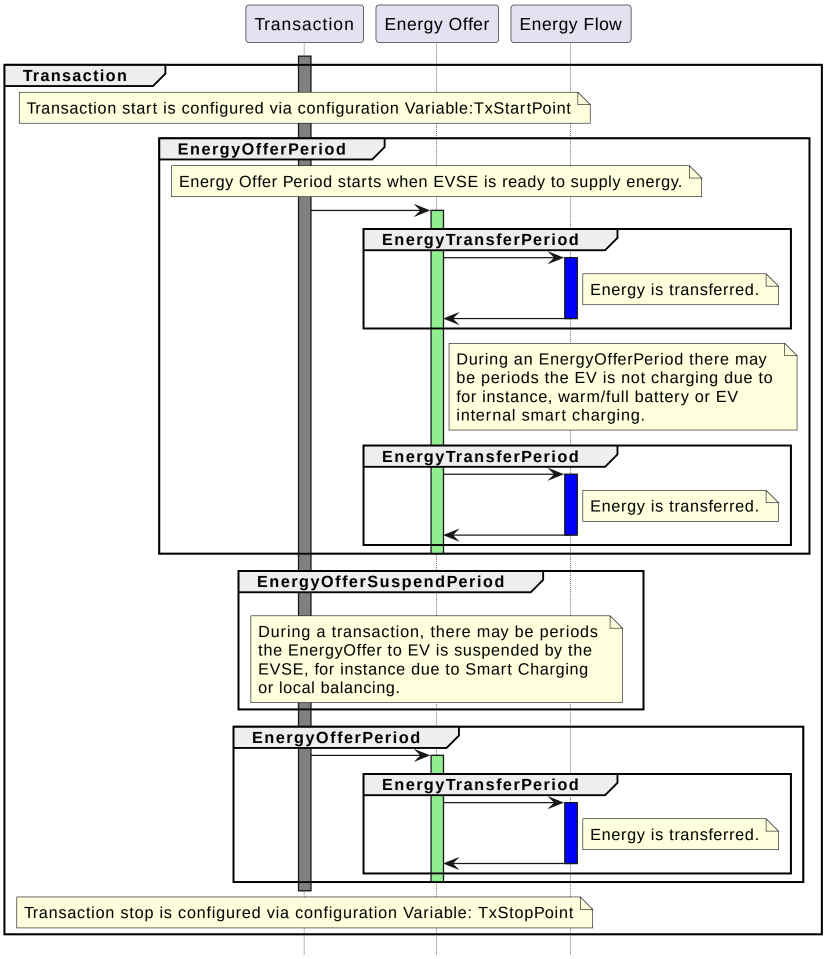

Transaction
Energy Offer
Energy Flow
Transaction
Transaction start is configured via configuration Variable:TxStartPoint
EnergyOfferPeriod
Energy Offer Period starts when EVSE is ready to supply energy.
EnergyTransferPeriod
Energy is transferred.
During an EnergyOfferPeriod there may be periods the EV is not charging due to for instance, warm/full battery or EV internal smart charging.
EnergyTransferPeriod
Energy is transferred.
EnergyOfferSuspendPeriod
During a transaction, there may be periods the EnergyOffer to EV is suspended by the EVSE, for instance due to Smart Charging or local balancing.
EnergyOfferPeriod
EnergyTransferPeriod
Energy is transferred.
Transaction stop is configured via configuration Variable: TxStopPoint

Figure 1. OCPP Charging Transaction definition

2.7. ISO 15118 support

This section is informative.

OCPP supports ISO 15118 authorization (also called "Plug and Charge") and ISO 15118 based Smart Charging. (See [ISO15118-2] and [ISO15118-20]). Furthermore, it describes how to install and update ISO 15118 certificates. This functionality is not included as one functional block, but is included in multiple chapters throughout the specification. ISO 15118 authorization is included in the functional block C. Authorization and the Smart Charging use cases for ISO 15118 are included in the section K 5.3. ISO 15118 based Smart Charging. Certificate handling is described in a M Certificate Management.

OCPP 2.1 has added specific support for bidirectional power transfer with ISO 15118-20 in Q. Bidirectional Power Transfer.

Implementors need to be aware of timeout constraints enforced by ISO 15118, see [ISO15118-2] (Page: 172, Table: 109) and [ISO15118-20] (Page 384, Table: 215).

For reference, the current timing constraints for ISO 15118-2:2014 are:

Table 9. ISO 15118-2 Timing constrains

Timeout	Default
Sequence Timeouts	60 seconds
Sequence Performance Timeouts	40 seconds
PaymentDetailsReq/Res	4.5 seconds
AuthorizationReq/Res	1.5 seconds
CertificateUpdateReq/Res	4.5 seconds
CertificateInstallationReq/Res	4.5 seconds

For reference, the current timing constraints for ISO 15118-20:2022 are:

Table 10. ISO 15118-20 Timing constrains

Timeout	Default
Sequence Timeouts	60 seconds
Sequence Performance Timeouts	40 seconds
AuthorizationSetupReq/Res	1.5 seconds
AuthorizationReq/Res	1.5 seconds
CertificateInstallationReq/Res	4.5 seconds

Chapter 3. Generic Requirements

This section is normative.

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

IMPORTANT This section requires knowledge of chapter 4 in Part 4 of the specification ([ref-ocpp20-part4]).

Table 11. 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 *statusInfo* element of the response message.
FR.03	When the Charging Station/CSMS receives an invalid OCPP request 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 rejected 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.	See use cases B02 and B03 for details.
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 (shown after the arrow "→") with the same *requestId* as the request: 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 for *requestId* = X was accepted, so that it can expect result messages for this *requestId*. TriggerMessage does not have a *requestId*, but the requirement still applies in the sense that a TriggerMessageResponse must be sent before the sending the requested message.
FR.06 **(2.1)**	When the Charging Station/CSMS receives an invalid OCPP response message according to the JSON schemas / RPC Framework	The Charging Station/CSMS SHALL respond with a RPC Framework: CALLRESULTERROR.	

FR.07 **(2.1)**	When the Charging Station sends an OCPP message RPC Framework type SEND	CSMS SHALL NOT respond with a message.	Receipt of a SEND message is not acknowledged. The sender will never know if the message has been received or processed.

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 constrains.

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

Chapter 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 2.1 supports three security profiles:
The table below shows which security measures are used by which profile.

Table 12. 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 2.1 without implementing security. While this is possible, it is NOT considered a valid OCPP 2.1 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 13. 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.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 recommended. The Charging Station SHALL only allow to lower the security profile if the variable AllowSecurityProfileDowngrade is implemented and set to *true*. In that case, the Charging Station SHALL only allow to downgrade from profile 3 to profile 2. The Charging Station SHALL NOT allow to downgrade from profile 2 or profile 3 to profile 1 using the OCPP protocol.
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 14. Security Profile 1 - Unsecured Transport with Basic Authentication

No.	Type	Description
1	**Name**	Unsecured Transport with Basic Authentication

No.	Type	Description
2	**Profile No.**	1
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.

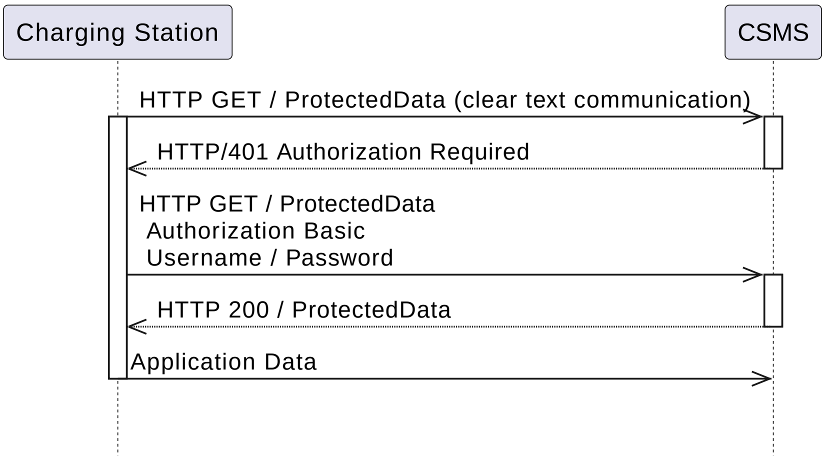


Figure 2. Sequence Diagram: HTTP Basic Authentication sequence diagram

7	**Remark(s)**	Please note, that the encoding of the basic authentication password in OCPP 2.1 (A00.FR.205) differs from how this was done in OCPP 1.6.

1.3.3. Unsecured Transport with Basic Authentication Profile - Requirements

Table 15. 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 a maximum as defined by the *maxLimit* of configuration variable `BasicAuthPassword`, which must be at least 40 characters and at most 64. 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 16. 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.

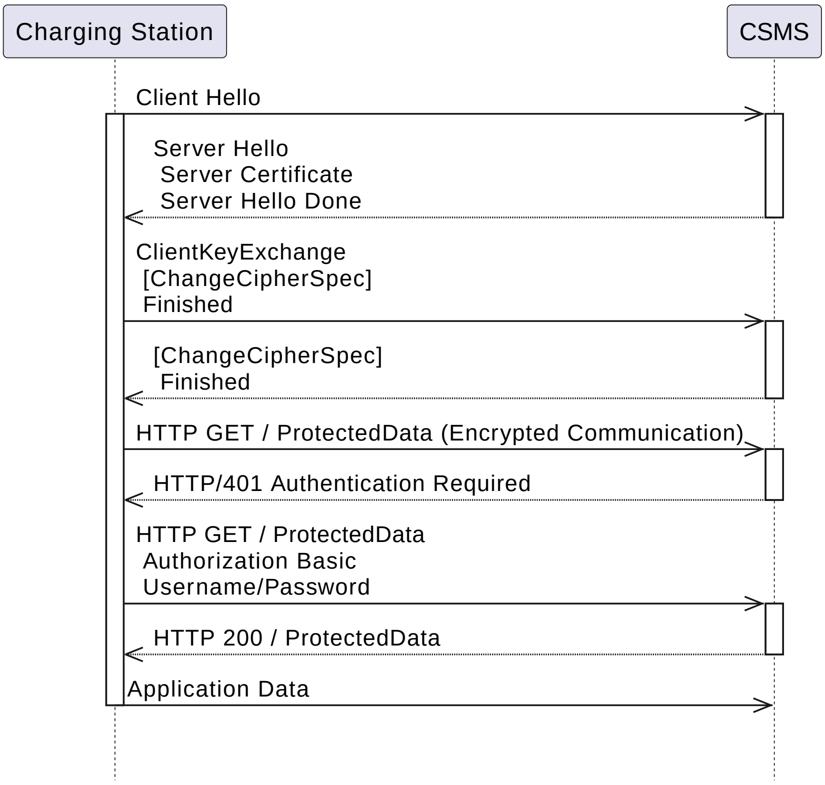


Figure 3. Sequence Diagram: TLS with Basic Authentication sequence diagram

7	**Remark(s)**	TLS allows a number of configurations, not all of which provide sufficient security. The requirements below describe the configurations allowed for OCPP. 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. **Example:** *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>)>* Please note, that the encoding of the basic authentication password in OCPP 2.1 (A00.FR.304) differs from how this was done in OCPP 1.6.

1.3.5. TLS with Basic Authentication Profile - Requirements

Table 17. 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] (Same as A00.FR.202)
A00.FR.302	A00.FR.301	The client, i.e. the Charging Station, SHALL provide a username and password with every connection request. (Same as A00.FR.203)
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. (Same as A00.FR.204)
A00.FR.304	A00.FR.302	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 a maximum as defined by the *maxLimit* of configuration variable `BasicAuthPassword`, which must be at least 40 characters and at most 64. The password SHALL be sent as a UTF-8 encoded string (NOT encoded into octet string or base64). (Same as A00.FR.205)
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 `commonName` 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 InvalidCsmsCertificate 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). 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. A security event only needs to be sent once for repeated failed connection attempts, in order to avoid overflow to the offline queue.
A00.FR.317		TLS SHALL be implemented as in [4] or its successor standards without any modifications.
A00.FR.318		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.
A00.FR.319		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**) Note 1: TLS_RSA does not support forward secrecy, therefore TLS_ECDHE is RECOMMENDED. In certain jurisdictions forward secrecy is mandatory. 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). Note 2: Please note that ISO15118-2 prescribes to implement the following cipher suites for the communication between EV and Charging Station: TLS_ECDH_ECDSA_WITH_AES_128_**CBC**_SHA256, TLS_ECDHE_ECDSA_WITH_AES_128_**CBC**_SHA256
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 18. 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.

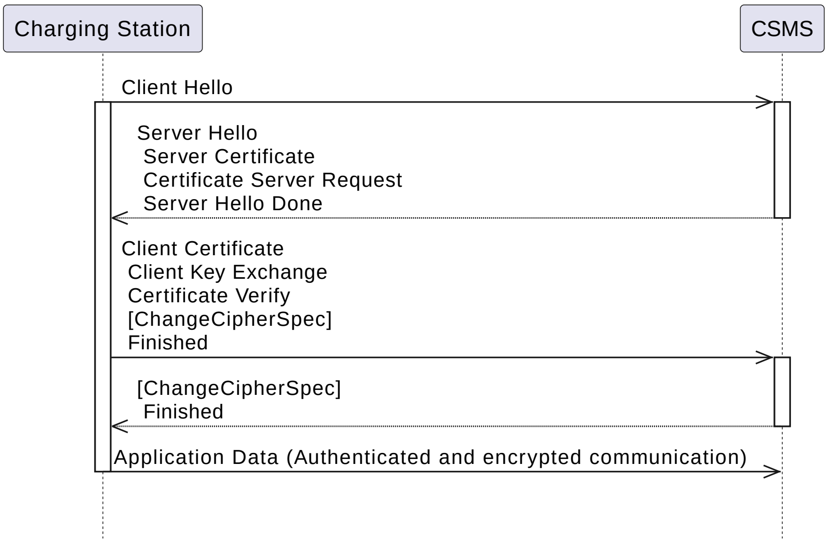


Figure 4. Sequence Diagram: TLS with Client Side Certificates

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

1.3.7. TLS with Client Side Certificates Profile - Requirements

Table 19. 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. (Same as A00.FR.306)
A00.FR.410		The CSMS SHALL authenticate itself by using the CSMS certificate as server side certificate. (Same as A00.FR.307)
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]. (Same as A00.FR.308)
A00.FR.412		The Charging Station SHALL verify that the `commonName` matches the CSMS's FQDN. (Same as A00.FR.309)
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). (Same as A00.FR.310)
A00.FR.414	A00.FR.413	The Charging Station SHALL terminate the connection. (Same as A00.FR.311)
A00.FR.415		The communication channel SHALL be secured using Transport Layer Security (TLS) [4]. (Same as A00.FR.312)
A00.FR.416		The Charging Station and CSMS SHALL only use TLS v1.2 or above. (Same as A00.FR.313)
A00.FR.417		Both of these endpoints SHALL check the version of TLS used. (Same as A00.FR.314)
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. (Same as A00.FR.315)
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). (Same as A00.FR.316) 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. A security event only needs to be sent once for repeated failed connection attempts, in order to avoid overflow to the offline queue.
A00.FR.420		TLS SHALL be implemented as in [4] or its successor standards without any modifications. (Same as A00.FR.317)

ID	Precondition	Requirement definition
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** (Same as A00.FR.318) 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.
A00.FR.422		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**) (Same as A00.FR.319) Note 1: TLS_RSA does not support forward secrecy, therefore TLS_ECDHE is RECOMMENDED. In certain jurisdictions formward secrecy is mandatory. 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). Note 2: Please note that ISO15118-2 prescribes to implement the following cipher suites for the communication between EV and Charging Station: TLS_ECDH_ECDSA_WITH_AES_128_**CBC**_SHA256, TLS_ECDHE_ECDSA_WITH_AES_128_**CBC**_SHA256
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. (Same as A00.FR.320)
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]. (Same as A00.FR.321)
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. (Same as A00.FR.322)
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). (Same as A00.FR.323)
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.

ID	Precondition	Requirement definition
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.
A00.FR.430	If the Charging Station certificate has expired	The Charging Station SHOULD still attempt to establish a connection with the CSMS and leave the decision to accept the connection up to the CSMS.

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 20. 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 21. 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.

ID	Precondition	Requirement definition
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 needs to meet the requirements in ISO 15118-2 or ISO 15118-20.
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 22. 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.

NOTE It is not recommended to have preinstalled well-known root CA certificates on a Charging Station like in operating systems or browsers, like for example a CA bundle. Only root and intermediate certificates part of the Charging Station Operator hierarchy should be used for the OCPP connection, as described by section Certificate Hierarchy. Trusting many additional well-known root CA certificates creates security risks.

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 connection can stay open longer than 24 hours.

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

Table 23. Recommended revocation methods for the different certificates.

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

Table 24. 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.
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 should becomes compromised, attackers could used 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 25. 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.

Chapter 2. Use cases & Requirements

A01 - Update Charging Station Password for HTTP Basic Authentication

No.	Type	Description
1	**Name**	Update Charging Station Password for HTTP Basic Authentication
2	**ID**	A01
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	**1.** The CSMS sends a SetVariablesRequest(ComponentName=SecurityCtrlr, VariableName=BasicAuthPassword) to the Charging Station. **2.** The Charging Station responds with SetVariablesResponse and the status *Accepted*. **3.** The Charging Station disconnects its current connection. (Storing any queued messages) **4.** The Charging Station connects to the CSMS with the new password.
5	**Prerequisite(s)**	Security Profile: Basic Security Profile or TLS with Basic Authentication in use.
6	**Postcondition(s)**	**Successful postcondition:** The Charging Station has reconnected to the CSMS with the new password. **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.
7	**Error handling**	n/a
8	**Remark(s)**	n/a

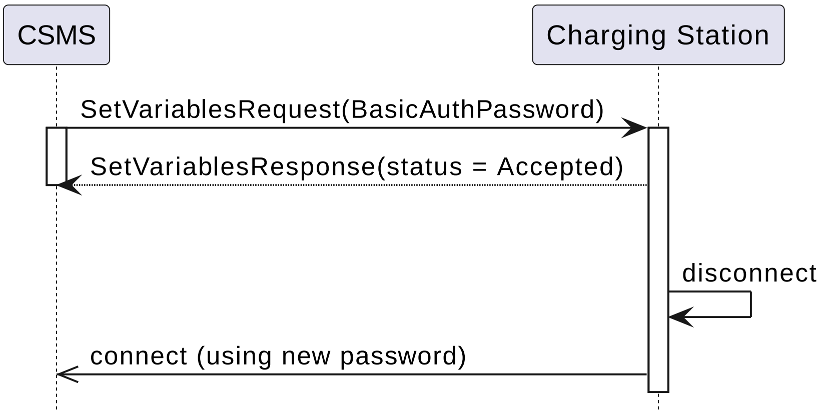

CSMS
Charging Station
SetVariablesRequest(BasicAuthPassword)
SetVariablesResponse(status = Accepted)
disconnect
connect (using new password)

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

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

Table 26. A01 - Requirements

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

ID	Precondition	Requirement definition
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 accepts 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

Updated in OCPP 2.1

No.	Type	Description
1	**Name**	Update Charging Station Certificate by request of CSMS
2	**ID**	A02
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/SignV2G20Certificate` for separate ISO 15118 certificate). If the Charging Station has a separate ISO15118Ctrlr (SECC in ISO 15118) for each EVSE, then CSMS will have to send a request for each of them. The device model the Charging Station will tell if ISO15118Ctrlr is located at toplevel or EVSE-level. If the Charging Station has multiple SECCs that each control multiple EVSEs, then these are represented in device model by an ISO15118Ctrlr for each EVSE. The EVSEs that are controlled by the same SECC report an ISO15118Ctrlr with the same "SeccId".
	Actors	Charging Station, CSMS, Certificate Authority Server

No.	Type	Description
	Scenario description	*SignChargingStationCertificate* **1.** The CSMS requests the Charging Station to update its certificate using the TriggerMessageRequest with the *requestedMessage* field set to SignChargingStationCertificate. **2.** The Charging Station responds with TriggerMessageResponse **3.** The Charging Station generates a new public / private key pair. **4.** The Charging Station sends a SignCertificateRequest to the CSMS containing the *certificateType* = `ChargingStationCertificate` . **5.** The CSMS responds with SignCertificateResponse, with status *Accepted*. **6.** The CSMS forwards the CSR to the Certificate Authority Server. **7.** Certificate Authority Server signs the certificate. **8.** The Certificate Authority Server returns the Signed Certificate to the CSMS. **9.** The CSMS sends CertificateSignedRequest to the Charging Station. **10.** The Charging Station verifies the Signed Certificate. **11.** The Charging Station responds with CertificateSignedResponse to the CSMS with the status *Accepted* or *Rejected*.
	Alternative scenario	*SignV2GCertificate* **1.** CSMS requests information about component ISO15118Ctrlr by sending a GetReportRequest for *componentVariable.component* = "ISO15118Ctrlr" and *componentVariable.variable* = "SeccId". **2.** For each **unique** SeccId that is returned: **2.1.** The CSMS requests the Charging Station to update its certificate using the TriggerMessageRequest with the *requestedMessage* field set to SignV2GCertificate for a 15118 certificate, and *evse* set to the EVSE of the ISO15118Ctrlr. (If ISO15118Ctrlr only exists as one component at toplevel, then *evse* can be omitted.) **2.2.** The Charging Station responds with TriggerMessageResponse **2.3.** The Charging Station generates a new public / private key pair. **2.4.** The Charging Station sends a SignCertificateRequest to the CSMS containing the *certificateType* = `V2GCertificate` and a *csr* in which the CommonName (CN) is set to the value of SeccId and with a *hashRootCertificate* element that identifies the Certificate Authority to use. **2.5.** CSMS responds with SignCertificateResponse, with status *Accepted*. **2.6.** The CSMS forwards the CSR to the Certificate Authority Server. **2.7.** Certificate Authority Server signs the certificate. **2.8.** The Certificate Authority Server returns the Signed Certificate to the CSMS. **2.9.** The CSMS sends CertificateSignedRequest to the Charging Station. **2.10.** The Charging Station verifies the Signed Certificate. **2.11.** The Charging Station responds with CertificateSignedResponse to the CSMS with the status *Accepted* or *Rejected*.
5	**Prerequisite(s)**	The standard configuration variable "OrganizationName" MUST be set. For SignV2GCertificate the variable ISO15118Ctrlr.SeccId 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.
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 be known. 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.

No.	Type	Description
8	Remark(s)	The Charging Station Operator may act as a certificate authority for the Charging Station Operator hierarchy. 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. 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. 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. ISO 15118-20 uses higher security algorithms than ISO 15118-2. A Charging Station that supports ISO 15118-20 therefore needs to install two SeccLeafCertificates to support both signature algorithms. In that case CSMS needs to request Charging Station via two TriggerMessageRequests to send a SignCertificateRequest for the ISO 15118-2 certificate and for the ISO 15118-20 certificate. This is reflected in requirements A02.FR.22/23.

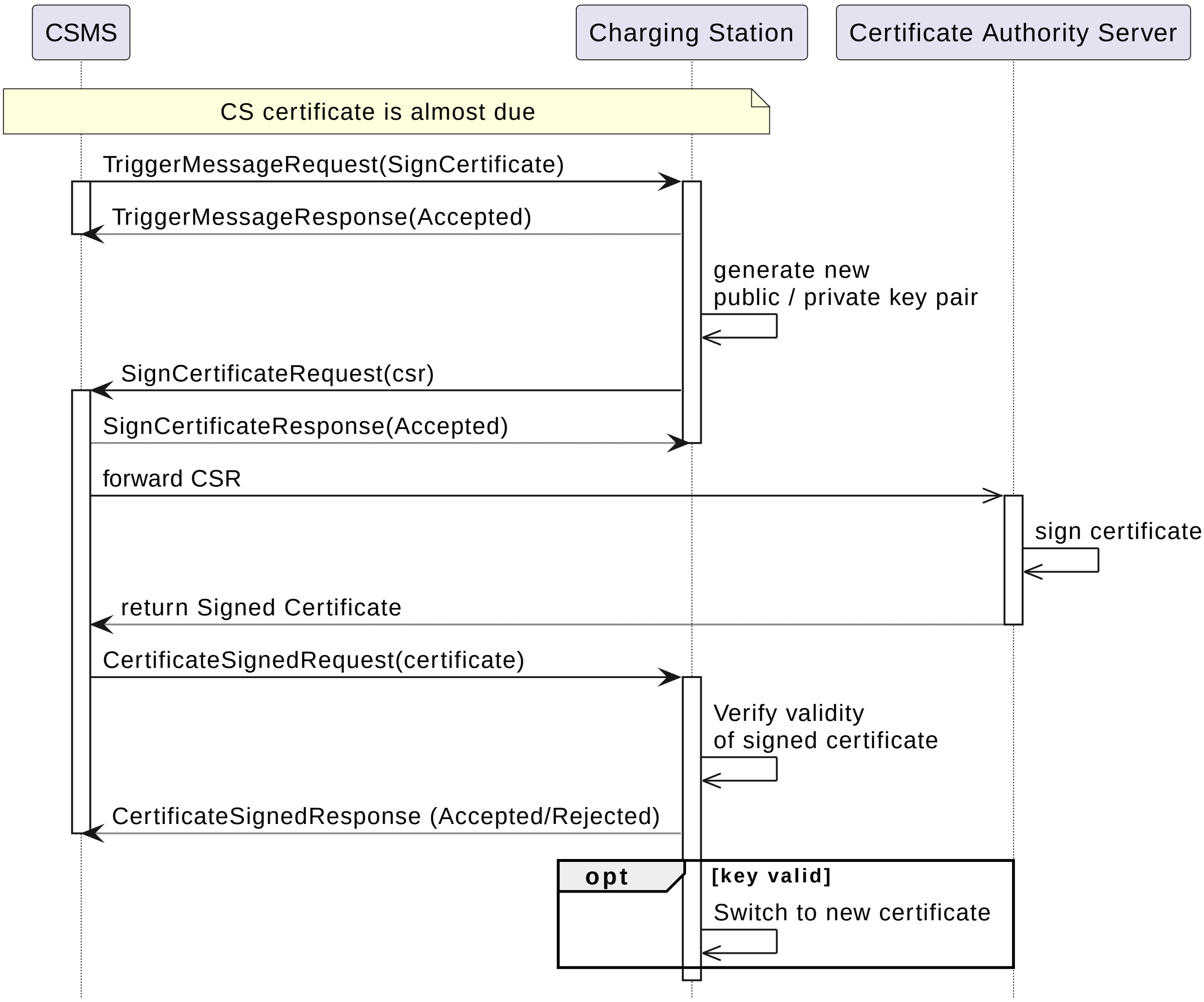

CSMS
Charging Station
Certificate Authority Server
CS certificate is almost due
TriggerMessageRequest(SignCertificate)
TriggerMessageResponse(Accepted)
generate new
public / private key pair
SignCertificateRequest(csr)
SignCertificateResponse(Accepted)
forward CSR
sign certificate
return Signed Certificate
CertificateSignedRequest(certificate)
Verify validity
of signed certificate
CertificateSignedResponse (Accepted/Rejected)
opt
[key valid]
Switch to new certificate

Figure 6. Update Charging Station Certificate

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

Table 27. A02 - Requirements

ID	Precondition	Requirement definition	Note
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 that the current date (at the time of the update) is within the certificate's validity period, the properties in Certificate Properties, and that it is part of the Charging Station Operator certificate hierarchy as described in Certificate Hierarchy.	When providing a newly signed client certificate with a start period that equals the current time, the CSMS should take into account that there might be a slight discrepancy in the time between the Charging Station and CSMS. This could cause the Charging Station to reject the new certificate, because in case a small time difference exists, the validity period might (just) be in the future for the device.
A02.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).	
A02.FR.08	If the certificate is valid.	The Charging Station SHALL respond to the CertificateSignedRequest with status Accepted AND the Charging Station SHALL switch to the new certificate by reconnecting the websocket and TLS connection.	
A02.FR.10	(A02.FR.08 OR A02.FR.28) AND The Charging Station successfully connected to the CSMS using either one of the certificates.	The Charging Station SHALL discard the client certificate that is NOT in use.	This is to prevent having multiple client certificates installed at the Charging Station, which the CSMS is unable to manage.
A02.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.	

ID	Precondition	Requirement definition	Note
A02.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.	
A02.FR.13	When using different certificates for 15118 connections and the Charging Station to CSMS connection	The Charging Station SHALL set the *certificateType* field in the SignCertificateRequest to the certificate for which the update was triggered.	
A02.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.	
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 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 MAY respond with a CertificateSignedResponse message with status *Rejected* .	
A02.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.	
A02.FR.18	A02.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.	
A02.FR.19 **(2.1)**	A02.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`, `SignV2G20Certificate` or `SignCombinedCertificate`.	
A02.FR.20 **(2.1)**	A02.FR.07	The Charging Station SHALL NOT initiate the back-off mechanism and resend the SignCertificateRequest, until this is requested by the CSMS via a TriggerMessageRequest for `SignChargingStationCertificate`, `SignV2GCertificate`, `SignV2G20Certificate` or `SignCombinedCertificate`.	

ID	Precondition	Requirement definition	Note
A02.FR.21	When the Charging Station receives a SignCertificateResponse with status `Rejected`, in response to a SignCertificateRequest with *certificateType* `V2GCertificate`	It is RECOMMENDED to turn off ISO15118PnCEnabled until the Charging Station has been rebooted.	
A02.FR.22 **(2.1)**	A02.FR.02 AND *requestedMessage* = `SignV2GCertificate` in TriggerMessageRequest	Charging Station SHALL send a SignCertificateRequest message for a V2G certificate for ISO 15118-2.	
A02.FR.23 **(2.1)**	A02.FR.02 AND *requestedMessage* = `SignV2G20Certificate` in TriggerMessageRequest	Charging Station SHALL send a SignCertificateRequest message for a V2G certificate for ISO 15118-20.	
A02.FR.24 **(2.1)**		Charging Station SHOULD add a *requestId* to SignCertificateRequest to be able to match the request to the resulting CertificateSignedRequest message.	
A02.FR.25 **(2.1)**	When CSMS receives a SignCertificateRequest with a field *requestId*	CSMS SHALL include this *requestId* in the resulting CertificateSignedRequest message.	
A02.FR.26 **(2.1)**	When Charging Station receives a CertificateSignedRequest with an unknown *requestId*	Charging Station SHALL respond with a CertificateSignedResponse with *status* = `Rejected`.	
A02.FR.27 **(2.1)**	When *certificateType* in SignCertificateRequest is `SignV2GCertificate` or `SignV2G20Certificate`	Charging Station SHOULD include *hashRootCertificate* in SignCertificateRequest with the certificate hash of the Root CA to identify the Certificate Authority to use.	
A02.FR.28	A02.FR.08 AND the charging station was not able to successfully connect to any of the configured entries of `NetworkConfigurationPriority` using the new certificate AND The Charging Station supports either one or both reconnection mechanisms described at requirements; B10.FR.07 and B10.FR.08.	The Charging Station SHALL for the reconnection mechanism described at B10.FR.07 fallback to the `old` client certificate AND for the reconnection mechanism described at B10.FR.08 alternate between using the `old` and new client certificate after all `NetworkConfigurationPriority` entries.	As described by requirement B10.FR.09, the Charging Station SHOULD NOT stop trying to reconnect to the CSMS. This is to prevent the Charging Station from becoming a stranded asset.
A02.FR.29	A02.FR.10 AND The Charging Station discarded the new client certificate.	The Charging Station SHOULD send a SecurityEventNotification *DiscardedRenewedClientCertificate* to the CSMS.	Otherwise the CSMS is not aware that the Charging Station discarded the new client certificate and the CSMS should again trigger a client certificate renewal.

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
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 ISO 15118-2 or `V2G20Certificate` for ISO 15118-20) 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 *certificateType* field.
	Actors	Charging Station, CSMS, Certificate Authority Server
	Scenario description	**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 `Accepted`. **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 `Accepted` or `Rejected`.
5	**Prerequisite(s)**	The standard configuration variable `OrganizationName` 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.
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 be known. 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.
8	**Remark(s)**	Same remarks as in A02 - Update Charging Station Certificate by request of CSMS apply.

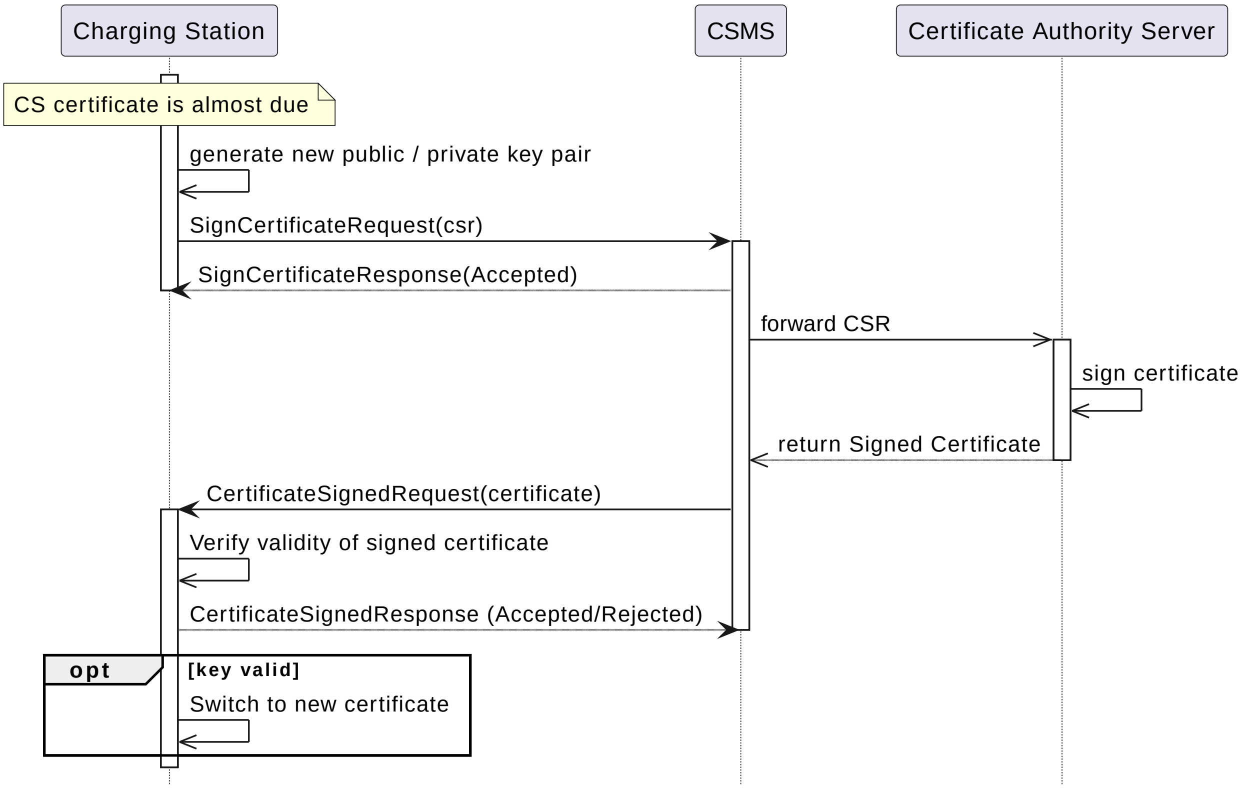

Charging Station
CSMS
Certificate Authority Server
CS certificate is almost due
generate new public / private key pair
SignCertificateRequest(csr)
SignCertificateResponse(Accepted)
forward CSR
sign certificate
return Signed Certificate
CertificateSignedRequest(certificate)
Verify validity of signed certificate
CertificateSignedResponse (Accepted/Rejected)
opt
[key valid]
Switch to new certificate

Figure 7. Update Charging Station Certificate initiated by Charging Station

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

Table 28. A03 - Requirements

ID	Precondition	Requirement definition	Note
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. (Same as A02.FR.03)	
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. (Same as A02.FR.04)	
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. (Same as A02.FR.05)	

ID	Precondition	Requirement definition	Note
A03.FR.06		The Charging Station SHALL verify the validity of the signed certificate in the CertificateSignedRequest message, checking that the current date (at the time of the update) is within the certificate's validity period, the properties in Certificate Properties, and that it is part of the Charging Station Operator certificate hierarchy as described in Certificate Hierarchy. (Same as A02.FR.06)	When providing a newly signed client certificate with a start period that equals the current time, the CSMS should take into account that there might be a slight discrepancy in the time between the Charging Station and CSMS. This could cause the Charging Station to reject the new certificate, because in case a small time difference exists, the validity period might (just) be in the future for the device.
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). (Same as A02.FR.07)	
A03.FR.08	If the certificate is valid.	The Charging Station SHALL respond to the CertificateSignedRequest with status Accepted AND the Charging Station SHALL switch to the new certificate by reconnecting the websocket and TLS connection. (Same as A02.FR.08)	
A03.FR.10	(A03.FR.08 OR A03.FR.24) AND The Charging Station successfully connected to the CSMS using either one of the certificates.	The Charging Station SHALL discard the client certificate that is NOT in use. (Same as A02.FR.10)	This is to prevent having multiple client certificates installed at the Charging Station, which the CSMS is unable to manage.
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. (Same as A02.FR.11)	
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. (Same as A02.FR.12)	
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. (Same as A02.FR.13)	
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. (Same as A02.FR.14)	
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. (Same as A02.FR.15)	

ID	Precondition	Requirement definition	Note
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 MAY respond with a CertificateSignedResponse message with status *Rejected* . (Same as A02.FR.16)	
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. (Same as A02.FR.17)	
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. (Same as A02.FR.18)	
A03.FR.19 **(2.1)**	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`, `SignV2G20Certificate` or `SignCombinedCertificate`. (Same as A02.FR.19)	
A03.FR.20 **(2.1)**		Charging Station SHOULD add a *requestId* to SignCertificateRequest to be able to match the request to the resulting CertificateSignedRequest message. (Same as A02.FR.24)	
A03.FR.21 **(2.1)**	When CSMS receives a SignCertificateRequest with a field *requestId*	CSMS SHALL include this *requestId* in the resulting CertificateSignedRequest message. (Same as A02.FR.25)	
A03.FR.22 **(2.1)**	When Charging Station receives a CertificateSignedRequest with an unknown *requestId*	Charging Station SHALL respond with a CertificateSignedResponse with *status* = `Rejected`. (Same as A02.FR.26)	
A03.FR.23 **(2.1)**	When *certificateType* in SignCertificateRequest is `SignV2GCertificate` or `SignV2G20Certificate`	Charging Station SHOULD include *hashRootCertificate* in SignCertificateRequest with the certificate hash of the Root CA to identify the Certificate Authority to use.	

ID	Precondition	Requirement definition	Note
A03.FR.24	A03.FR.08 AND the charging station was not able to successfully connect to any of the configured entries of `NetworkConfigurationPriority` using the new certificate AND The Charging Station supports either one or both reconnection mechanisms described at requirements; B10.FR.07 and B10.FR.08.	The Charging Station SHALL for the reconnection mechanism described at B10.FR.07 fallback to the `old` client certificate AND for the reconnection mechanism described at B10.FR.08 alternate between using the `old` and new client certificate after all `NetworkConfigurationPriority` entries. (Same as A02.FR.28)	As described by requirement B10.FR.09, the Charging Station SHOULD NOT stop trying to reconnect to the CSMS. This is to prevent the Charging Station from becoming a stranded asset.
A03.FR.25	A03.FR.10 AND The Charging Station discarded the new client certificate.	The Charging Station SHOULD send a SecurityEventNotification *DiscardedRenewedClientCertificate* to the CSMS. (Same as A02.FR.29)	Otherwise the CSMS is not aware that the Charging Station discarded the new client certificate and the CSMS may need to trigger a new client certificate renewal.

A04 - Security Event Notification

No.	Type	Description
1	**Name**	Security Event Notification
2	**ID**	A04
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	**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.
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*)

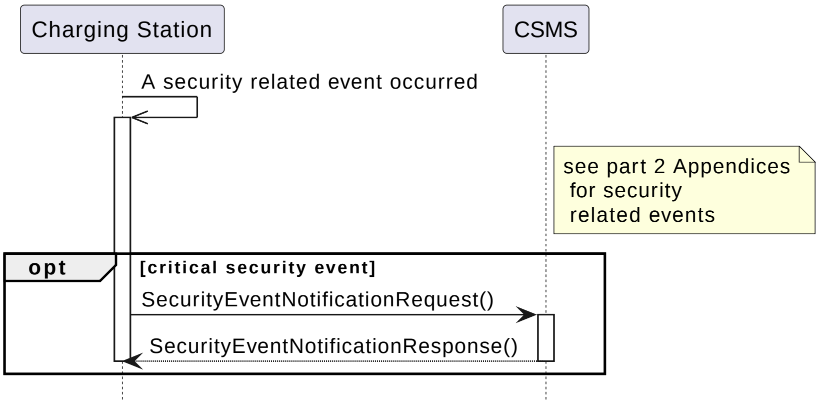

Charging Station
CSMS
A security related event occurred
see part 2 Appendices for security related events
opt
[critical security event]
SecurityEventNotificationRequest()
SecurityEventNotificationResponse()

Figure 8. Security Event Notification

A04 - Security Event Notification - Requirements

Table 29. A04 - Requirements

ID	Precondition	Requirement definition	Note
A04.FR.01	When a *critical* 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

No.	Type	Description
1	**Name**	Upgrade Charging Station Security Profile
2	**ID**	A05
3	**Objective(s)**	The CSO wants to change 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 2.1. Before migrating to a security profile, the prerequisites, like installed certificates or password, need to be configured.
	Actors	CSMS, Charging Station
	Scenario description	**1.** The CSMS sets a new value for the `NetworkConfigurationPriority` Configuration Variable via SetVariablesRequest, such that the NetworkConnectionProfile for the new security profile becomes first in the list and the existing connection profile becomes second in the list. **2.** The Charging Station responds with a SetVariablesResponse with status *Accepted* **3.** The CSMS sends a ResetRequest(OnIdle) **4.** The Charging Station reboots and connects via the new primary NetworkConnectionProfile
5	**Prerequisite(s)**	The CSO ensures that a NetworkConnectionProfile has been set using an allowed security profile AND that the prerequisite(s) for going to the new security profile are met before sending the command to change to the new security profile.
6	**Postcondition(s)**	The Charging Station was successfully upgraded to a new security profile.
7	**Error handling**	n/a
8	**Remark(s)**	For security reasons it is by default not allowed to revert to a lower Security Profile using OCPP. Only when the variable AllowSecurityProfileDowngrade is implemented and is set to true, it is allowed to downgrade from profile 3 to profile 2. Even in that case, it is not allowed to revert from profile 2 or profile 3 to security profile 1 using OCPP.

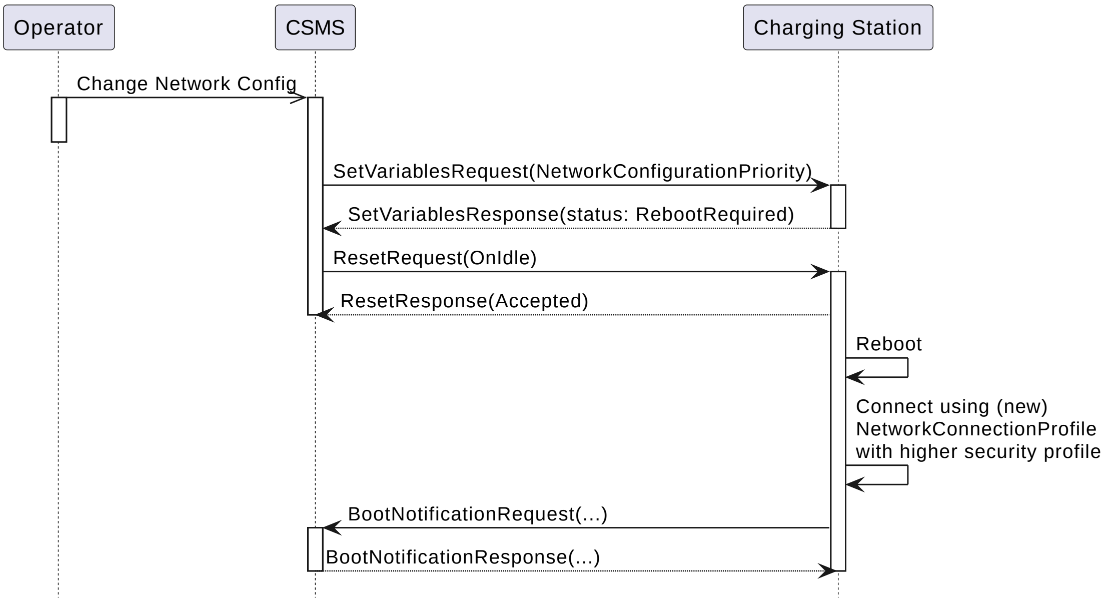

Operator
CSMS
Charging Station
Change Network Config
SetVariablesRequest(NetworkConfigurationPriority)
SetVariablesResponse(status: RebootRequired)
ResetRequest(OnIdle)
ResetResponse(Accepted)
Reboot
Connect using (new)
NetworkConnectionProfile
with higher security profile
BootNotificationRequest(...)
BootNotificationResponse(...)

Figure 9. Upgrade Charging Station Security Profile

A05 - Upgrade Charging Station Security Profile - Requirements

Table 30. A05 - Requirements

ID	Precondition	Requirement definition
A05.FR.02	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	The Charging Station SHALL respond with SetVariablesResponse(Rejected), and not update the value for `SecurityProfile` and/or reconnect to the CSMS.
A05.FR.03	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	The Charging Station SHALL respond with SetVariablesResponse(Rejected), and not update the value for `SecurityProfile` and/or reconnect to the CSMS.
A05.FR.04	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	The Charging Station SHALL respond with SetVariablesResponse(Accepted)
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	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.
A05.FR.07	A05.FR.06	The CSMS SHALL NOT allow the Charging Station to connect with a lower security profile anymore.
A05.FR.08	The variable AllowSecurityProfileDowngrade is implemented and set to true AND The currently active 'SecurityProfile' is 3 AND The Charging Station receives SetVariablesRequest for NetworkConfigurationPriority containing profile slots for NetworkConnectionProfiles with a 'securityProfile' value equal to 2.	The Charging Station SHALL respond with SetVariablesResponse(Accepted)
A05.FR.09	The variable AllowSecurityProfileDowngrade is implemented and set to true AND The currently active 'SecurityProfile' is higher than 1 AND The Charging Station receives SetVariablesRequest for NetworkConfigurationPriority containing profile slots for NetworkConnectionProfiles with a 'securityProfile' value equal to 1.	The Charging Station SHALL respond with SetVariablesResponse(Rejected)
A05.FR.10	The variable AllowSecurityProfileDowngrade is not implemented or implemented and set to false AND The Charging Station receives SetVariablesRequest for NetworkConfigurationPriority containing profile slots for NetworkConnectionProfiles with a 'securityProfile' value lower than the currently active security profile	The Charging Station SHALL respond with SetVariablesResponse(Rejected)

B. Provisioning

Chapter 1. Introduction

This Functional Block describes the functionality that helps a CSO provision their Charging Stations: permitting them on their network, retrieving configuration information from these Charging Stations, making changes to their 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`.

Chapter 2. Use cases & Requirements

2.1. Booting a Charging Station

B01 - Cold Boot Charging Station

No.	Type	Description
1	**Name**	Cold Boot Charging Station
2	**ID**	B01
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	**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.** *Optional:* The Charging Station sends NotifyEventRequest with component.name *Connector*, variable.name *AvailabilityState* and actualValue *Unavailable* to the CSMS for each Connector. **5.** The Charging Station sends NotifyEventRequest with component.name *Connector*, variable.name *AvailabilityState* to the CSMS for each Connector. If the *AvailabilityState* was set to *Unavailable* or *Reserved* from the CSMS prior to the (re)boot, the Connector should return to this *AvailabilityState*, otherwise the *AvailabilityState* should be *Available* or, when it resumes a transaction that was ongoing, the *AvailabilityState* should be *Occupied*. **6.** Normal operation 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)**	**Successful postcondition:** The Charging Station is in *Idle* status, and *Accepted*. **Failure postcondition:** The Charging Station received the status *Rejected*, B03 - Cold Boot Charging Station -Rejected applies. The Charging Station received the status *Pending*, B02 - Cold Boot Charging Station - Pending applies.
7	**Error handling**	**1.** No initial establishment of connection of communication between the CSMS and Charging Station: Retry Connection with the CSMS. **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 dit not get a BootNotificationResponse containing this interval), in a way that avoids flooding the CSMS with requests.
8	**Remark(s)**	Multiple options for a self check are possible: some Charging Stations boot and send status notifications with *Unavailable*, then perform a check of all the hardware and send new NotifyEvents with AvailabilityState *Available* 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 *before* sending a BootNotificationRequest and determine the status before a (mobile) network connection is established and a BootNotificationRequest is sent. When something is wrong with the Charging Station or EVSE, the status SHALL be set to *Faulted*. *Reserved* and *Unavailable* states persist after a reboot.

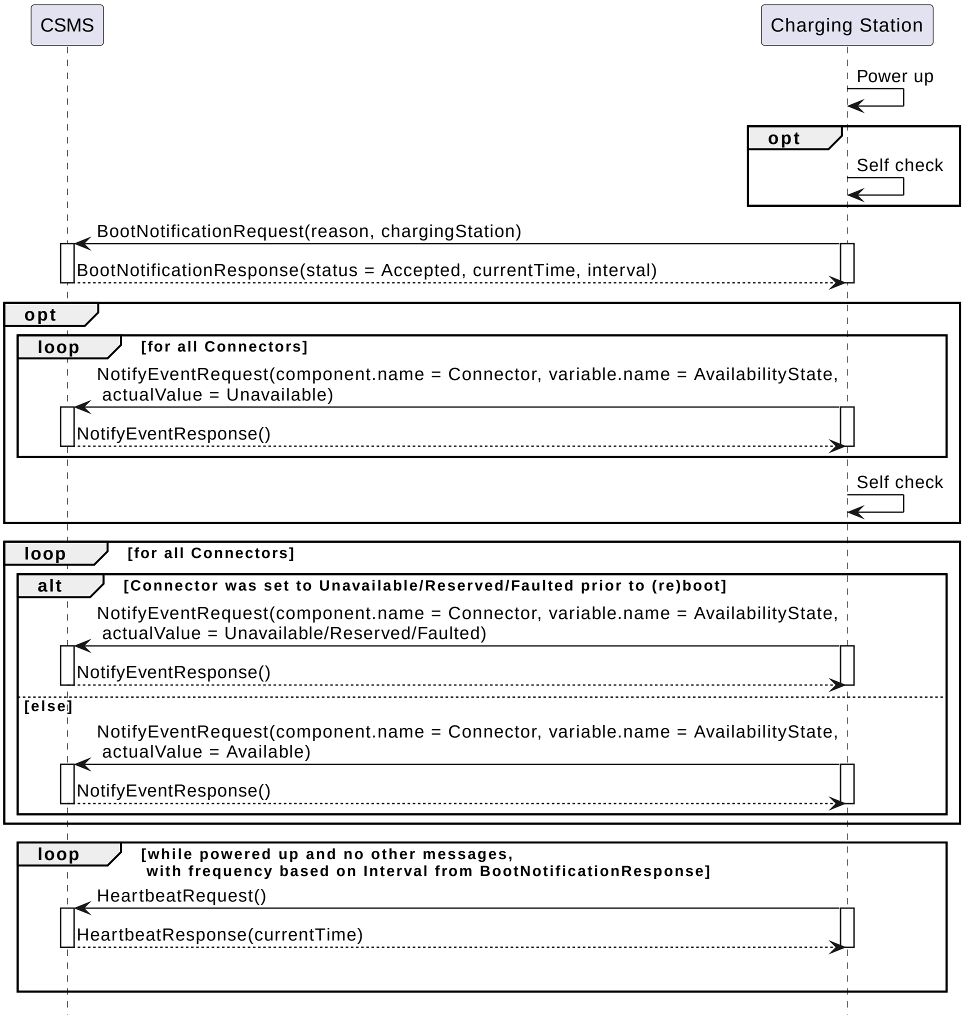

CSMS
Charging Station
Power up
opt
Self check
BootNotificationRequest(reason, chargingStation)
BootNotificationResponse(status = Accepted, currentTime, interval)
opt
loop [for all Connectors]
NotifyEventRequest(component.name = Connector, variable.name = AvailabilityState, actualValue = Unavailable)
NotifyEventResponse()
Self check
loop [for all Connectors]
alt [Connector was set to Unavailable/Reserved/Faulted prior to (re)boot]
NotifyEventRequest(component.name = Connector, variable.name = AvailabilityState, actualValue = Unavailable/Reserved/Faulted)
NotifyEventResponse()
[else]
NotifyEventRequest(component.name = Connector, variable.name = AvailabilityState, actualValue = Available)
NotifyEventResponse()
loop [while powered up and no other messages, with frequency based on Interval from BootNotificationResponse]
HeartbeatRequest()
HeartbeatResponse(currentTime)

Figure 10. Sequence Diagram: Cold Boot Charging Station

B01 - Cold Boot Charging Station - Requirements

Table 31. 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.	
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 **(2.1)**	When the CSMS responds with BootNotificationResponse with the status `Accepted`.	The Charging Station SHALL send a NotifyEventRequest with *variable.name* = "AvailabilityState" for each Connector with its current state.	Use of StatusNotificationRequest instead of NotifyEventRequest is still allowed, but StatusNotificationRequest is deprecated.

ID	Precondition	Requirement definition	Note
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 `Unavailable` by a Change Availability command.	The `Unavailable` 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 `Pending` status.
B01.FR.09	B01.FR.01	The Charging Station SHALL indicate the reason for sending the BootNotificationRequest message in the *reason* 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 `Reserved` state MUST be persistent across reboots.	

B02 - Cold Boot Charging Station - Pending

No.	Type	Description
1	**Name**	Cold Boot Charging Station - Pending
2	**ID**	B02
	Parent use case	B01 - Cold Boot Charging Station
3	**Objective(s)**	**1.** To inform the Charging Station that it is not yet accepted by the CSMS: *Pending* 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 *Pending* status.
	Actors	Charging Station, CSMS
	Scenario description	**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 *Pending*. **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)**	**1.** The CSMS requires to set the Charging Station in *Pending* status. **2.** The Charging Station is starting up (i.e. powering up after being powered down).
6	**Postcondition(s)**	**Successful postcondition:** The Charging Station is in *Pending* status. **Failure postcondition:** The Charging Station received the status *Rejected*, B03 - Cold Boot Charging Station -Rejected applies.
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.

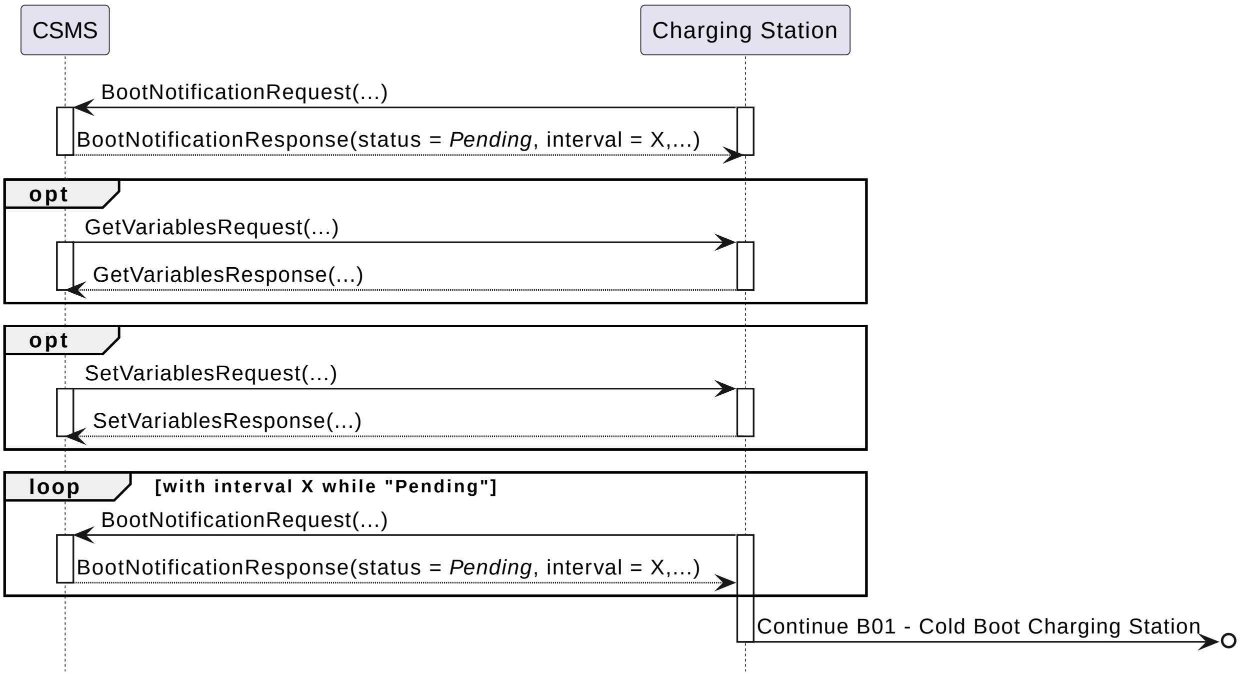

CSMS
Charging Station
BootNotificationRequest(...)
BootNotificationResponse(status = Pending, interval = X,...)
opt
GetVariablesRequest(...)
GetVariablesResponse(...)
opt
SetVariablesRequest(...)
SetVariablesResponse(...)
loop
[with interval X while "Pending"]
BootNotificationRequest(...)
BootNotificationResponse(status = Pending, interval = X,...)
Continue B01 - Cold Boot Charging Station

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

B02 - Cold Boot Charging Station - Pending - Requirements

Table 32. B02 - Requirements

ID	Precondition	Requirement definition	Note
B02.FR.01	After the Charging Station received the *Pending* 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 *Pending* status AND receiving a RequestStartTransactionRequest or RequestStopTransactionRequest	The Charging Station SHALL respond with a RequestStartTransactionResponse or RequestStopTransactionResponse with status *Rejected*. (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.	
B02.FR.09	The Charging Station has received a BootNotificationResponse with status *Pending* 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.

B03 - Cold Boot Charging Station - Rejected

No.	Type	Description
1	**Name**	Cold Boot Charging Station - Rejected
2	**ID**	B03
	Parent use case	B01 - Cold Boot Charging Station
3	**Objective(s)**	To inform the Charging Station that its *not* (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 *Rejected* status.
	Actors	Charging Station, CSMS
	Scenario description	**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 *Rejected* 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)**	**1.** The CSMS requires to set the Charging Station in the *Rejected* status. **2.** The Charging Station is powered down.
6	**Postcondition(s)**	The Charging Station remains in the *Rejected* status.
7	**Error handling**	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.
8	**Remark(s)**	During the status *Rejected*, 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. Additionally, the CSMS MAY close the communication channel, for instance to free up system resources. It is advised *not* 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 When the CSMS returns with BootNotificationResponse with the status *Accepted*, B01 - Cold Boot Charging Station applies.

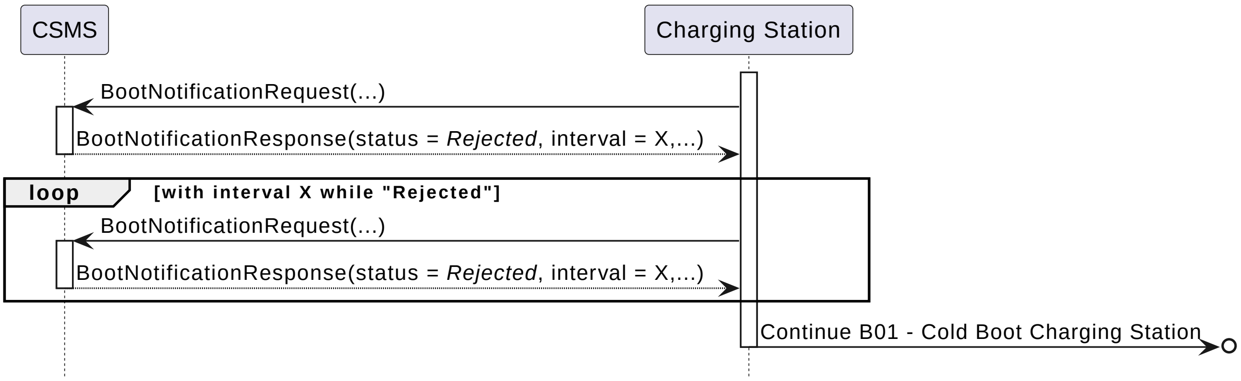

CSMS
Charging Station
BootNotificationRequest(...)
BootNotificationResponse(status = Rejected, interval = X,...)
loop
[with interval X while "Rejected"]
BootNotificationRequest(...)
BootNotificationResponse(status = Rejected, interval = X,...)
Continue B01 - Cold Boot Charging Station

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

B03 - Cold Boot Charging Station - Rejected - Requirements

Table 33. 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.

ID	Precondition	Requirement definition
B03.FR.02	If the CSMS returns the status *Rejected*. 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	When the CSMS has Rejected the BootNotificationRequest from the Charging Station.	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 passed.
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 response to a BootNotificationRequest from Charging Station	Charging Station SHALL respond with RPC Framework: CALLERROR: SecurityError.

B04 - Offline Behavior Idle Charging Station

No.	Type	Description
1	**Name**	Offline Behavior Idle Charging Station
2	**ID**	B04
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 *Offline*.
	Actors	Charging Station, CSMS
	Scenario description	**1.** The CSMS or communication is unavailable. **2.** The Charging Station operates stand-alone. **3.** The connection is restored. **4.** If the *Offline* period exceeds the value of the OfflineThreshold Configuration Variable: the Charging Station sends a NotifyEventRequest with variable.name *AvailabilityState* to the CSMS for each connector. Otherwise it only sends a NotifyEventRequest with variable.name *AvailabilityState* 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 *Offline* behavior, the CSMS knows the Charging Stations' and EVSEs' state.
7	**Error handling**	The offline situation is a 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

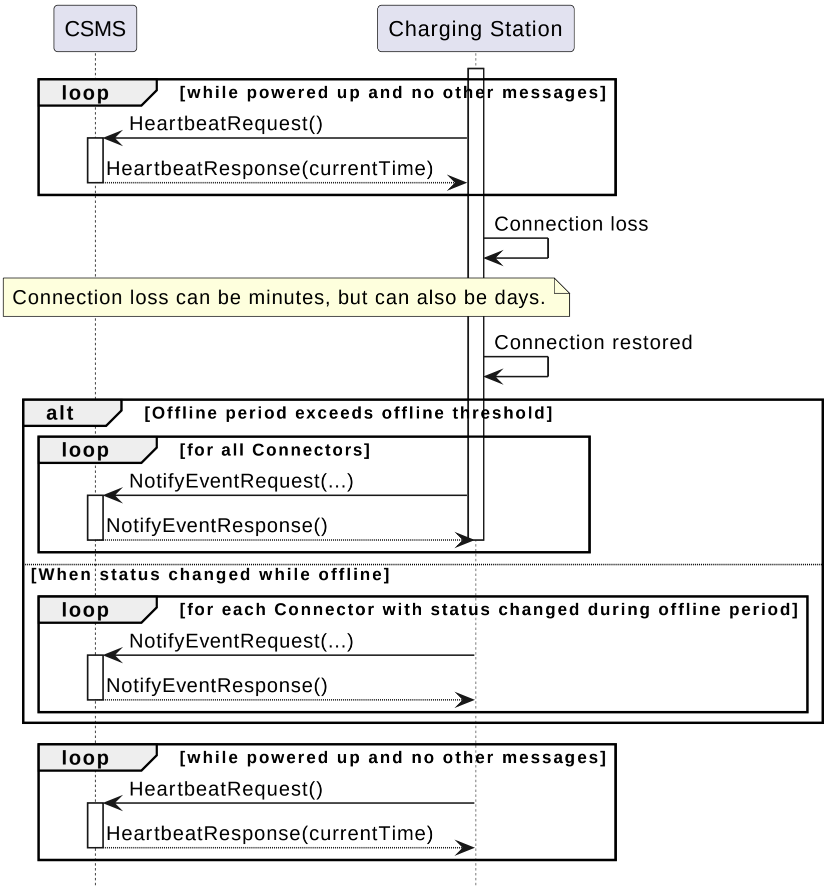

CSMS
Charging Station
loop [while powered up and no other messages]
HeartbeatRequest()
HeartbeatResponse(currentTime)
Connection loss
Connection loss can be minutes, but can also be days.
Connection restored
alt [Offline period exceeds offline threshold]
loop [for all Connectors]
NotifyEventRequest(...)
NotifyEventResponse()
[When status changed while offline]
loop [for each Connector with status changed during offline period]
NotifyEventRequest(...)
NotifyEventResponse()
loop [while powered up and no other messages]
HeartbeatRequest()
HeartbeatResponse(currentTime)

Figure 13. Sequence Diagram: Offline Behavior Idle Charging Station

B04 - Offline Behavior Idle Charging Station - Requirements

Table 34. B04 - Requirements

ID	Precondition	Requirement definition
B04.FR.01 **(2.1)**	After having been *Offline* AND the *Offline* period exceeds the value of the OfflineThreshold Configuration Variable.	The Charging Station SHALL send NotifyEventRequest with variable.name *AvailabilityState* to report the current status of all its Connectors.
B04.FR.02 **(2.1)**	After having been *Offline* AND the *Offline* period does NOT exceed the value of the OfflineThreshold Configuration Variable.	The Charging Station SHALL send NotifyEventRequest with variable.name *AvailabilityState* 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 2.1: Part 1 - Architecture & Topology", chapter 4.

B05 - Set Variables

No.	Type	Description
1	**Name**	Set Variables
2	**ID**	B05
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
	Scenario description	**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 executed the change(s).
5	**Prerequisite(s)**	n/a
6	**Postcondition(s)**	**Successful postconditions:** **1.** The change was executed *Successfully*. **Failure postconditions:** **1.** The variable is supported, but setting could not be changed, the Charging Station responds with the status *Rejected*. **2.** The variable is *not* supported, the Charging Station responds with the status *UnknownVariable*.
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.

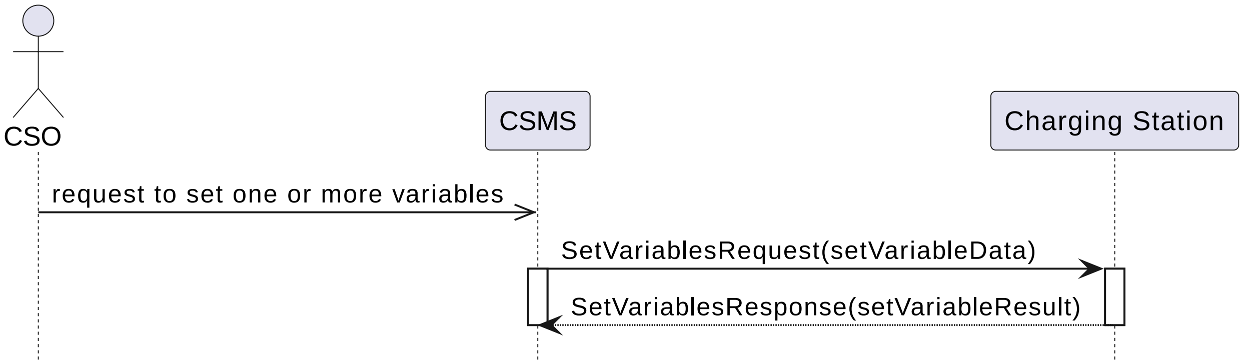

CSO
CSMS
Charging Station
request to set one or more variables
SetVariablesRequest(setVariableData)
SetVariablesResponse(setVariableResult)

Figure 14. Sequence Diagram: Set Variables

B05 - Set Variables - Requirements

Table 35. 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 *component* and *variable* combination as one of the SetVariableData elements in the SetVariablesRequest.
B05.FR.03	B05.FR.02 AND If the SetVariablesRequest contains an *attributeType*	The corresponding SetVariableResult element in the SetVariablesResponse SHALL also contain the same *attributeType*
B05.FR.04	When the Charging Station receives a SetVariablesRequest with an unknown Component in the SetVariableData	The Charging Station SHALL set the *attributeStatus* field in the corresponding SetVariableResult to: UnknownComponent.
B05.FR.05	When the Charging Station receives a SetVariablesRequest with a Variable that is unknown for the given Component in the SetVariableData	The Charging Station SHALL set the *attributeStatus* field in the corresponding SetVariableResult to: UnknownVariable.
B05.FR.06	When the Charging Station receives a SetVariablesRequest with an attributeType that is unknown for the given Variable in the SetVariableData	The Charging Station SHALL set the *attributeStatus* field in the corresponding SetVariableResult to: NotSupportedAttributeType.
B05.FR.07	When the Charging Station receives a SetVariablesRequest with a *value* that is incorrectly formatted for the given Variable in the SetVariableData	The Charging Station SHALL set the *attributeStatus* field in the corresponding SetVariableResult to: Rejected. (More information can be provided in the optional *statusInfo* element.)
B05.FR.08	When the Charging Station receives a SetVariablesRequest with a *value* that is lower or higher than the range of the given Variable in the SetVariableData	The Charging Station SHALL set the *attributeStatus* field in the corresponding SetVariableResult to: Rejected. (More information can be provided in the optional *statusInfo* element.)
B05.FR.09	NOT (B05.FR.04 to B05.FR.08) AND When the Charging Station receives a SetVariablesRequest for a Variable in the SetVariableData, but is not able to set it	The Charging Station SHALL set the *attributeStatus* field in the corresponding SetVariableResult to: Rejected. (This happens if the variable is *ReadOnly*, 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 *value* from the SetVariableData	The Charging Station SHALL set the *attributeStatus* 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 *attributeType*.	The corresponding SetVariableResult element in the SetVariablesResponse SHALL contain the *attributeType* Actual.

ID	Precondition	Requirement definition
B05.FR.13		The CSMS SHALL NOT include multiple SetVariableData elements, in a single SetVariablesRequest, with the same Component, Variable and *AttributeType* combination. Note that an omitted *AttributeType* counts as the value *Actual*.

B06 - Get Variables

No.	Type	Description
1	**Name**	Get Variables
2	**ID**	B06
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	**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)**	**Successful postcondition:** The Charging Station was able to send all the requested variables. **Failure postcondition:** The Charging Station was not able to send all requested variables.
7	**Error handling**	n/a
8	**Remark(s)**	n/a

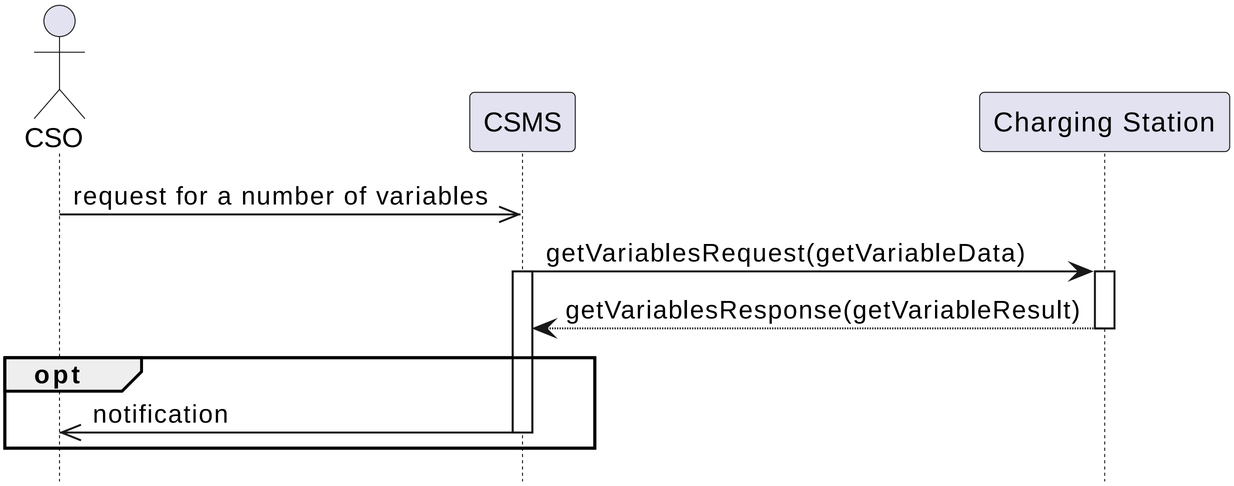

CSO
CSMS
Charging Station
request for a number of variables
getVariablesRequest(getVariableData)
getVariablesResponse(getVariableResult)
opt
notification

Figure 15. Sequence Diagram: Get Variables

B06 - Get Variables - Requirements

Table 36. 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 *component* and *variable* combination as one of the GetVariableData elements in the GetVariablesRequest.
B06.FR.03	B06.FR.02 AND If the GetVariablesRequest contains an *attributeType*	The corresponding GetVariableResult element in the GetVariablesResponse SHALL also contain the same *attributeType*

ID	Precondition	Requirement definition
B06.FR.04	B06.FR.01	Every GetVariableResult element in the GetVariablesResponse SHALL contain an *attributeValue* with the value of an attribute from the requested *attributeType* 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 *attributeStatus* field in the corresponding GetVariableResult to: UnknownComponent AND SHALL omit the *attributeValue*.
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 *attributeStatus* field in the corresponding GetVariableResult to: UnknownVariable AND SHALL omit the *attributeValue*.
B06.FR.08	When the Charging Station receives a GetVariablesRequest with an attributeType that is unknown for the given Variable in the GetVariableData	The Charging Station SHALL set the *attributeStatus* field in the corresponding GetVariableResult to: NotSupportedAttributeType AND SHALL omit the *attributeValue*.
B06.FR.09	When the Charging Station receives a GetVariablesRequest for a Variable in the GetVariableData that is *WriteOnly*	The Charging Station SHALL set the *attributeStatus* field in the corresponding GetVariableResult to: Rejected.
B06.FR.10	When the Charging Station was able to get the *value* requested from a GetVariablesRequest	The Charging Station SHALL set the *attributeStatus* field in the corresponding GetVariableResult to: Accepted and set the *attributeValue* to the found value.
B06.FR.11	When the Charging Station receives a GetVariablesRequest without an *attributeType*.	The corresponding GetVariableResult element in the GetVariablesResponse SHALL contain the *attributeType* Actual.
B06.FR.13	NOT B06.FR.08 AND the Charging Station has no *attributeValue* for the requested *attributeType* of the componentvariable	Charging Station SHALL return an empty string as *attributeValue*. Note: this can happen, for example, when the *attributeType* `Target` has not yet been set, even though it is supported.
B06.FR.14	B06.FR.01 AND a value for *instance* is provided in the *component* and/or *variable* 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 *instance* in the *component* and/or *variable* in GetVariableData AND a component and/or variable without an *instance* does not exist	Charging Station SHALL return the *attributeStatus* `UnknownComponent` or `UnknownVariable` 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(OccurenceConstraintViolation)
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(FormatViolation)

B07 - Get Base Report

No.	Type	Description
1	**Name**	Get Base Report
2	**ID**	B07
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	**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)**	**Successful postcondition:** The Charging Station was able to send the requested report. **Failure postcondition:** The Charging Station was *not* able to send the requested report.
7	**Error handling**	n/a
8	**Remark(s)**	n/a

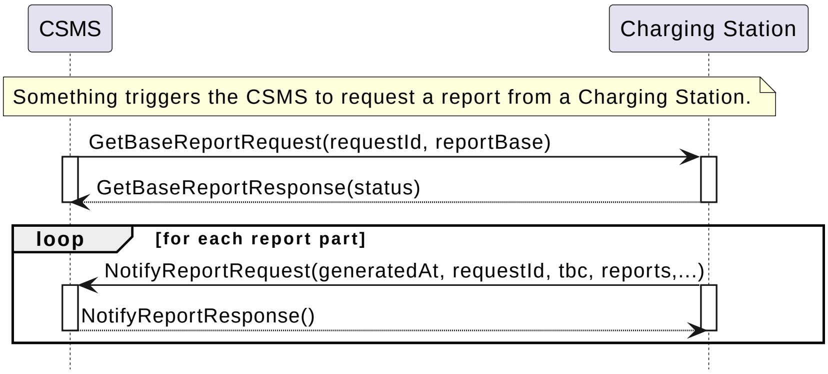

CSMS
Charging Station
Something triggers the CSMS to request a report from a Charging Station.
GetBaseReportRequest(requestId, reportBase)
GetBaseReportResponse(status)
loop
[for each report part]
NotifyReportRequest(generatedAt, requestId, tbc, reports,...)
NotifyReportResponse()

Figure 16. Sequence Diagram: Get Base Report

B07 - Get Base Report - Requirements

Table 37. B07 - Requirements

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

ID	Precondition	Requirement definition	Note
B07.FR.03	B07.FR.01	The Charging Station SHALL send the requested information, excluding the *value* of WriteOnly variables, via one or more NotifyReportRequest messages to the CSMS.	It is good practice to send the report data in as few messages as possible in order to limit data overhead.
B07.FR.04	B07.FR.01 AND The getBaseReportRequest contained a *requestId*	Every NotifyReportRequest send for this getBaseReportRequest SHALL contain the same *requestId*.	
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 *reportBase* 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 *VariableCharacteristics*.	
B07.FR.08	B07.FR.01 AND When *reportBase* is FullInventory	Then the Charging Station SHALL respond with a NotifyReportRequest to report on all component-variables including their *VariableCharacteristics*.	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 *reportBase* 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 operationalStatus 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: - AvailabilityState. For each EVSE Component: - AvailabilityState. For each Connector Component: - AvailabilityState (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 seqNo field of the NotifyReportRequest is incremental per report. So the NotifyReportRequest message which contains the first report part, SHALL have a seqNo 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 *reportBase* = `FullInventory` 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.
B07.FR.15	When the Charging Station is sending the requested information via one or more NotifyReportRequest messages to the CSMS	The Charging Station SHALL omit the *value* of readonly variables	

B08 - Get Custom Report

No.	Type	Description
1	**Name**	Get Custom Report
2	**ID**	B08
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	**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)**	**Successful postcondition:** The Charging Station was able to send the requested report. **Failure postcondition:** The Charging Station was *not* able to send the requested report.
7	**Error handling**	n/a
8	**Remark(s)**	n/a

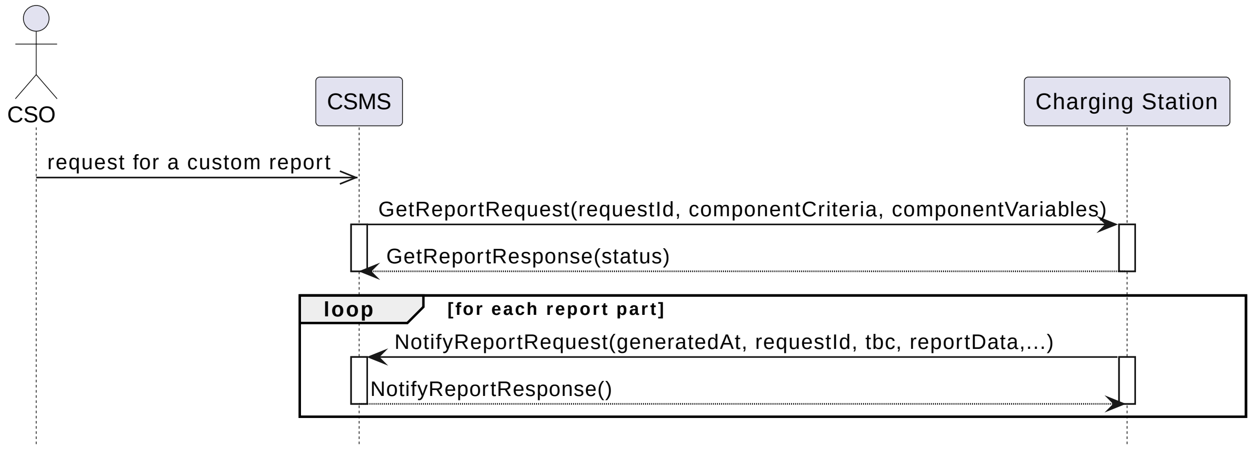

CSO
CSMS
Charging Station
request for a custom report
GetReportRequest(requestId, componentCriteria, componentVariables)
GetReportResponse(status)
loop
[for each report part]
NotifyReportRequest(generatedAt, requestId, tbc, reportData,...)
NotifyReportResponse()

Figure 17. Sequence Diagram: Get Custom Report

B08 - Get Custom Report - Requirements

Table 38. B08 - Requirements

ID	Precondition	Requirement definition
B08.FR.01	NOT B08.FR.15 AND NOT B08.FR.16 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, excluding the *value* of WriteOnly variables, via one or more NotifyReportRequest messages to the CSMS.

ID	Precondition	Requirement definition
B08.FR.04	B08.FR.01 AND The GetReportRequest contained a *requestId*	Every NotifyReportRequest sent for this GetReportRequest SHALL contain the same *requestId*.
B08.FR.05	B08.FR.01 AND *componentCriteria* and *componentVariables* are NOT both empty.	Every NotifyReportRequest sent for this GetReportRequest SHALL be limited to the set *componentCriteria* and *componentVariables*.
B08.FR.06		CSMS SHALL NOT send more *componentVariables* in one GetReportRequest message than is configured in the `ItemsPerMessageGetReport` Configuration Variable
B08.FR.07	B08.FR.01 AND *ComponentCriteria* contains: *Active*	The Charging Station SHALL report every component that has the variable *Active* set to *true*, or does not have the *Active* variable in a NotifyReportRequest.
B08.FR.08	B08.FR.01 AND *ComponentCriteria* contains: *Available*	The Charging Station SHALL report every component that has the variable *Available* set to *true*, or does not have the *Available* variable, in a NotifyReportRequest.
B08.FR.09	B08.FR.01 AND *ComponentCriteria* contains: *Enabled*	The Charging Station SHALL report every component that has the variable *Enabled* set to *true*, or does not have the *Enabled* variable, in a NotifyReportRequest.
B08.FR.10	B08.FR.01 AND *ComponentCriteria* contains: *Problem*	The Charging Station SHALL report every component that has the variable *Problem* set to *true* in a NotifyReportRequest.
B08.FR.11	B08.FR.01 AND *componentCriteria* is absent AND *componentVariables* is NOT empty.	Every NotifyReportRequest sent for this GetReportRequest is limited to the set in *componentVariables*.
B08.FR.12	B08.FR.01	The reported variables in NotifyReportRequest SHALL contain *variableCharacteristics*.
B08.FR.13	B08.FR.01 AND More than one *componentCriteria* 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 seqNo field of the NotifyReportRequest is incremental per report. So the NotifyReportRequest message which contains the first report part, SHALL have a seqNo 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 `Rejected`.
B08.FR.17	Charging Station receives a GetReportRequest with more ComponentVariableType elements than allowed by `ItemsPerMessageGetReport`	The Charging Station MAY respond with a CALLERROR(OccurenceConstraintViolation)
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.20	When Charging Station receives a GetReportRequest with *componentVariable* elements in which *variable* is missing	The Charging Station SHALL report for every *variable* of the *component* in *componentVariable*.
B08.FR.21	When Charging Station receives a GetReportRequest with *componentVariable* elements in which *variable* is present, but *instance* is missing	The Charging Station SHALL report for every instance of the *variable* of the *component* in *componentVariable*.
B08.FR.22	B08.FR.11 AND When Charging Station receives a GetReportRequest with a *component* in a *componentVariable* element that has a *component.evse.id*, but *component.evse.connector* is missing	The Charging Station SHALL report the component(s) with this *component.name*, *component.instance* and *component.evse.id* for every *component.evse.connector*, whilst taking into account B08.FR.24.

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

B09 - Setting a new NetworkConnectionProfile

(Updated in OCPP 2.1)

No.	Type	Description
1	**Name**	Setting a new NetworkConnectionProfile.
2	**ID**	B09
3	**Objective(s)**	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	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. For example: *valuesList* = "0,1,2" in the base report tells CSMS that three configuration slots, numbered 0, 1 and 2, are available (but not necessarily set). The configuration slot number that is used for the active connection is reported by variable OCPPCommCtrlr.ActiveNetworkProfile. 1. The CSMS sends a SetNetworkProfileRequest PDU containing an updated connection profile and a *configurationSlot* out of the *valuesList* 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`.
	Alternative scenario (New in OCPP 2.1)	*Setting network configuration via device model* The changing of network configuration parameters must be done for a configuration slot that is currently not listed in the NetworkConfigurationPriority variable. This ensures that Charging Station cannot use an incompletely configured network configuration. All parameters of a SetNetworkProfileRequest for a *configurationSlot* are present as device model variables on component NetworkConfiguration with instance = *configurationSlot* and are set directly via a SetVariablesRequest. See `NetworkConfiguration` for a list of variables. 1. If the configuration slot to be set is part of NetworkConfigurationPriority configuration variable a. CSMS will remove this configuration slot from the NetworkConfigurationPriority via a SetVariablesRequest, so that it can be updated. 2. The CSMS sends one or multiple SetVariablesRequest(s) to set new values for the variables in the NetworkConfiguration component with instance *configurationSlot*. 3. The Charging Station responds to each request with a SetVariablesResponse 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
7	**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
8	**Remarks**	When changes are made to the currently active NetworkConnectionProfile using SetNetworkProfileRequest, these will not be used until a reconnect occurs, which can be triggered by resetting a Charging Station as described in B10 - Migrate to new CSMS. The existing SetNetworkProfileRequest has become deprecated and may be subject to removal in a future OCPP release.

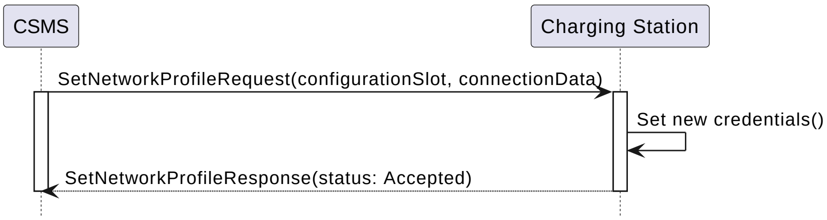

CSMS
Charging Station
SetNetworkProfileRequest(configurationSlot, connectionData)
Set new credentials()
SetNetworkProfileResponse(status: Accepted)

Figure 18. Sequence Diagram: Set Network Connection Profile

B09 - Requirements

Table 39. B09 - Requirements when using SetNetworkProfileRequest

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 *Accepted*
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 *Rejected*
B09.FR.03	If setting the new networkprofile fails.	The Charging Station SHALL respond by sending a SetNetworkProfileResponse message, with status *Failed*
B09.FR.04	The variable AllowSecurityProfileDowngrade is not implemented or implemented and set to false AND The Charging Station receives a SetNetworkProfileRequest AND the NetworkConnectionProfile contains a lower securityProfile than the currently active security profile	The Charging Station SHALL respond by sending a SetNetworkProfileResponse message, with status *Rejected*
B09.FR.05	When the value of *configurationSlot* in SetNetworkProfileRequest does not match an entry in *valuesList* of NetworkConfigurationPriority	The Charging Station SHALL respond by sending a SetNetworkProfileResponse message with status *Rejected*
B09.FR.06		A Charging Station SHALL support at least two configuration slots for network connection profiles.
B09.FR.07 **(2.1)**		Charging Station and CSMS are strongly RECOMMENDED to use the highest security profile possible.
B09.FR.08 **(2.1)**	When receiving a GetBaseReportRequest	Charging Station SHALL include in the report the component NetworkConfiguration with instances for every *configurationSlot* that it supports with the variables it supports.
B09.FR.09 **(2.1)**	When Charging Station accepts a SetNetworkProfileRequest from CSMS for *configurationSlot*	Charging Station SHALL update all corresponding NetworkConfiguration component variables with instance *configurationSlot*.
B09.FR.10 **(2.1)**	If Charging Station supports setting network configuration via configuration keys in NetworkConfiguration	Charging Station SHALL report these variables as writable (ReadWrite or WriteOnly in case of a password)
B09.FR.11 **(2.1)**	B09.FR.09 AND The SetNetworkProfileRequest does not contain the *apn* or the *vpn* field	Charging Station SHALL set variable ApnEnabled resp. VpnEnabled in NetworkConfiguration to False.
B09.FR.12 **(2.1)**	B09.FR.09 AND The SetNetworkProfileRequest contains the *apn* or the *vpn* field	Charging Station SHALL set variable ApnEnabled resp. VpnEnabled in NetworkConfiguration to True.
B09.FR.13 **(2.1)**	B09.FR.10 AND Charging Station does not support APN or VPN connections	Charging Station SHALL report the variable ApnEnabled resp. VpnEnabled in NetworkConfiguration as False and ReadOnly.
B09.FR.14 **(2.1)**		CSMS SHOULD use the SetNetworkProfileRequest to set the identity or BasicAuth password of the Charging Station. (See also B09.FR.26/27)
When using SetVariablesRequest for NetworkConfiguration component		

ID	Precondition	Requirement definition
B09.FR.20 **(2.1)**	B09.FR.10 AND On receipt of a SetVariablesRequest containing the variable NetworkConfigurationPriority AND the new value only removes configuration slot(s) from the current value	The Charging Station SHALL respond by sending a SetVariablesResponse with the corresponding *setVariableResult* containing status `Accepted`
B09.FR.21 **(2.1)**	B09.FR.10	CSMS SHALL NOT send a SetVariablesRequest containing any NetworkConfiguration component variable for which the component instance matches any of the members in the currently configured NetworkConfigurationPriority
B09.FR.22 **(2.1)**	B09.FR.10 AND On receipt of a SetVariablesRequest containing any NetworkConfiguration component variable AND the component instance matches any of the members in the currently configured NetworkConfigurationPriority	The Charging Station SHOULD respond by sending a SetVariablesResponse with the corresponding *setVariableResult* containing status `Rejected`
B09.FR.23 **(2.1)**	B09.FR.10 AND On receipt of a SetVariablesRequest containing any NetworkConfiguration component variable AND the component instance matches none of the members in the currently configured NetworkConfigurationPriority	The Charging Station SHALL validate the value of the variable.
B09.FR.24 **(2.1)**	B09.FR.23 AND variable validation was successful	The Charging Station SHALL store the value and respond by sending a SetVariablesResponse with the corresponding *setVariableResult* containing *status* `Accepted`
B09.FR.25 **(2.1)**	B09.FR.23 AND variable validation was not successful	The Charging Station SHALL respond by sending a SetVariablesResponse with the corresponding *setVariableResult* containing *status* `Rejected`
B09.FR.26 **(2.1)**	B09.FR.10 AND On receipt of a SetVariablesRequest containing the variable SecurityCtrlr.Identity AND the mutability of this variable is read/write	The Charging Station SHALL also set the variable of the same name in all NetworkConfiguration component instances to the same value (if valid), including component instances which are contained in the currently configured NetworkConfigurationPriority. This is for backwards compatibility only. CSMS SHOULD set the NetworkConfiguration component variable instead.
B09.FR.27 **(2.1)**	B09.FR.10 AND On receipt of a SetVariablesRequest containing the variable SecurityCtrlr.BasicAuthPassword	The Charging Station SHALL also set the variable of the same name in all NetworkConfiguration component instances to the same value (if valid), including component instances which are contained in the currently configured NetworkConfigurationPriority. This is for backwards compatibility only. CSMS SHOULD set the NetworkConfiguration component variable instead.
B09.FR.28 **(2.1)**	B09.FR.10 AND When Charging Station activates a new network configuration	Charging Station SHALL ensure that the values of SecurityCtrlr.Identity and SecurityCtrlr.BasicAuthPassword match the corresponding variables of NetworkConfiguration.Identity[*configurationSlot*] and NetworkConfiguration.BasicAuthPassword[*configurationSlot*] for the currently active *configurationSlot*.
B09.FR.29 **(2.1)**	When CSMS configures variables for an APN or VPN connection	CSMS SHALL set variable ApnEnabled resp. VpnEnabled of NetworkConfiguration to True.
B09.FR.30 **(2.1)**	When CSMS does not configure variables for an APN or VPN connection	CSMS SHALL set variable ApnEnabled resp. VpnEnabled of NetworkConfiguration to False.
B09.FR.31	The variable AllowSecurityProfileDowngrade is implemented and set to true AND The currently active 'SecurityProfile' is 3 AND The Charging Station receives a SetNetworkProfileRequest AND the NetworkConnectionProfile contains a securityProfile with a value of 2.	The Charging Station SHALL respond with SetVariablesResponse(Accepted)

ID	Precondition	Requirement definition
B09.FR.32	The variable AllowSecurityProfileDowngrade is implemented and set to true AND The currently active 'SecurityProfile' is higher than 1 AND The Charging Station receives a SetNetworkProfileRequest AND the NetworkConnectionProfile contains a securityProfile with a value of 1.	The Charging Station SHALL respond with SetVariablesResponse(Rejected)

B10 - Migrate to new CSMS

No.	Type	Description
1	**Name**	Migrate to new CSMS, using a different NetworkConnectionProfile.
2	**ID**	B10
3	**Objective(s)**	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	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. For example: *valuesList* = "0,1,2" in the base report tells CSMS that three configuration slots, numbered 0, 1 and 2, are available (but not necessarily set). The *value* of NetworkConfigurationPriority reports the order in which network configurations are tried to make a connection. For example: *value* = "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. **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 *Accepted* **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.
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.
7	**Error Handling**	n/a
8	**Remarks**	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. When an operator wants to perform an immediate switch, he should stop the transactions first.

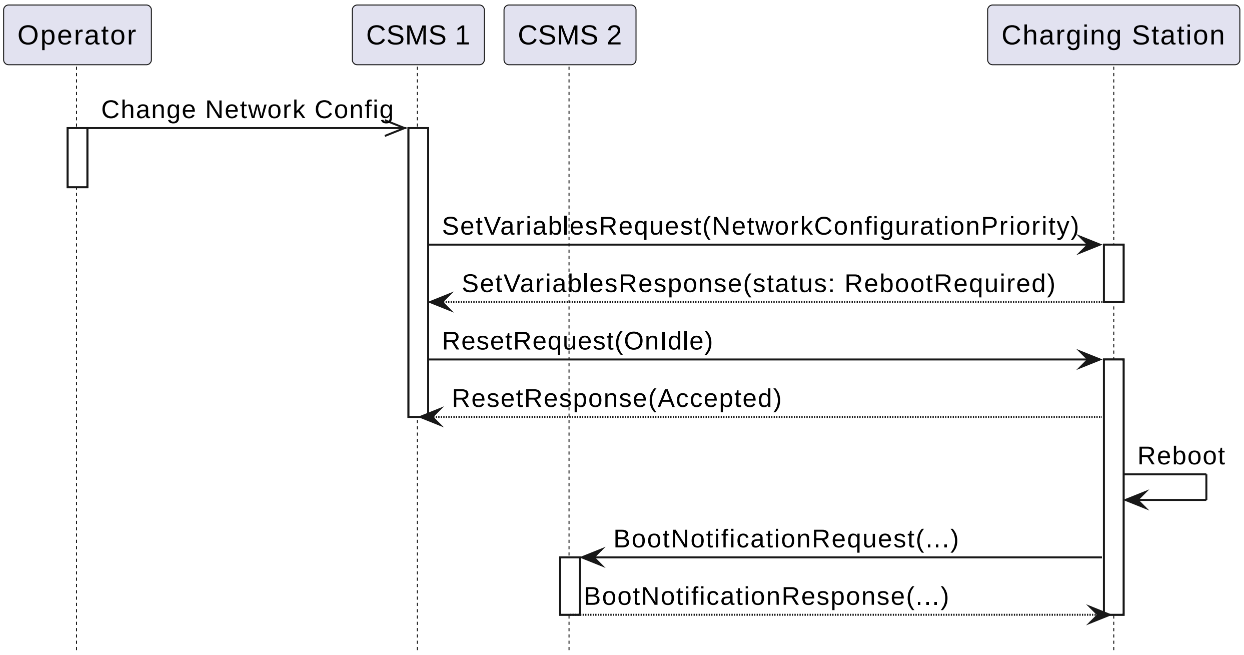

Operator
CSMS 1
CSMS 2
Charging Station
Change Network Config
SetVariablesRequest(NetworkConfigurationPriority)
SetVariablesResponse(status: RebootRequired)
ResetRequest(OnIdle)
ResetResponse(Accepted)
Reboot
BootNotificationRequest(...)
BootNotificationResponse(...)

Figure 19. Sequence Diagram: Migrate to new ConnectionProfile

B10 - Migrate to new NetworkConnectionProfile - Requirements

Table 40. 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`	Same as A05.FR.05
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.09	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".
B10.FR.08	B10.FR.09	The Charging Station SHOULD restart connecting with all configured entries of the `NetworkConfigurationPriority`	
B10.FR.09	B10.FR.03 AND All `NetworkProfileConnectionAttempts` for every entry of `NetworkConfigurationPriority` failed.	The Charging Station SHOULD NOT stop trying to reconnect to the CSMS. The Charging Station SHOULD implement the reconnecting mechanism described at requirement B10.FR.07 or B10.FR.08.	This is to prevent the Charging Station from becoming a stranded asset.

2.3. Resetting a Charging Station

B11 - Reset - Without Ongoing Transaction

No.	Type	Description
1	**Name**	Reset - Without Ongoing Transaction
2	**ID**	B11
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 paramater *evseId*, 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
	Scenario description	**1.** The CSO requests the CSMS to reset the Charging Station or EVSE. **2.** The CSMS sends ResetRequest requesting the Charging Station to reset itself or EVSE. **3.** The CSMS requests for an OnIdle or Immediate reset. **4.** The Charging Station responds with ResetResponse, indicating whether the Charging Station is able to reset itself or EVSE. **5.** The CSMS sends an optional notification to the CSO. **6.** Only if no evseId was supplied, then after the reset, the Charging Station will proceed as in use case B01.
	Alternative scenario(s)	B12 - Reset With Ongoing Transaction
5	**Prerequisite(s)**	No transaction is ongoing.
6	**Postcondition(s)**	**Successful postcondition:** The Charging Station was able to reset itself or EVSE. **Failure postcondition:** The Charging Station *not* was able to reset itself or EVSE.
7	**Error handling**	n.a
8	**Remark(s)**	Persistent states: for example, EVSE set to *Unavailable* SHALL persist. The Charging Station responds with ResetResponse.

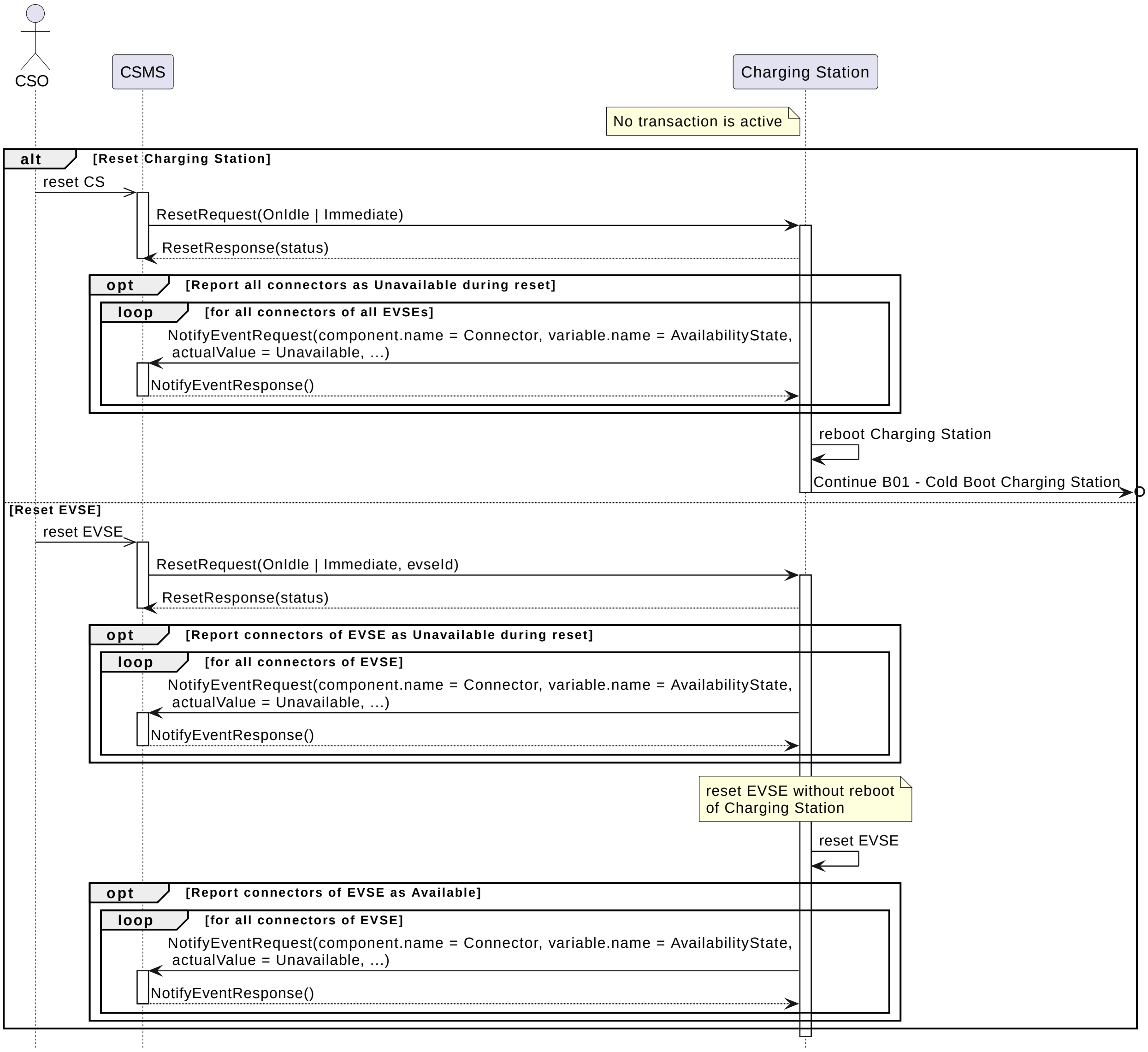

CSO
CSMS
Charging Station
No transaction is active
alt [Reset Charging Station]
reset CS
ResetRequest(OnIdle | Immediate)
ResetResponse(status)
opt [Report all connectors as Unavailable during reset]
loop [for all connectors of all EVSEs]
NotifyEventRequest(component.name = Connector, variable.name = AvailabilityState, actualValue = Unavailable, ...)
NotifyEventResponse()
reboot Charging Station
Continue B01 - Cold Boot Charging Station
[Reset EVSE]
reset EVSE
ResetRequest(OnIdle | Immediate, evseId)
ResetResponse(status)
opt [Report connectors of EVSE as Unavailable during reset]
loop [for all connectors of EVSE]
NotifyEventRequest(component.name = Connector, variable.name = AvailabilityState, actualValue = Unavailable, ...)
NotifyEventResponse()
reset EVSE without reboot of Charging Station
reset EVSE
opt [Report connectors of EVSE as Available]
loop [for all connectors of EVSE]
NotifyEventRequest(component.name = Connector, variable.name = AvailabilityState, actualValue = Unavailable, ...)
NotifyEventResponse()

Figure 20. Sequence Diagram: Reset Without Transaction

B11 - Reset - Without Ongoing Transaction - Requirements

Table 41. 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 *Inoperative* by the CSMS.	After a reboot of the Charging Station, the EVSEs SHALL return to the state *Unavailable* as prior to the reboot.
B11.FR.03	B11.FR.01 AND no *evseId* parameter is supplied AND ResetResponse has *status* = `Accepted`.	The Charging Station MAY send a NotifyEventRequest for *component* = "Connector", *variable* = "AvailabilityState", *actualValue* = "Unavailable" and *trigger* = "Delta" for each EVSE, 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.
B11.FR.05	If the status of an EVSE was *Reserved*.	After a reboot of the Charging Station or resetting of EVSE, the EVSE(s) SHALL return to the state *Reserved*.
B11.FR.06	B11.FR.01 AND Charging Station is at this moment not able to perform a reset.	The Charging Station SHALL respond with a *status* = `Rejected`.

ID	Precondition	Requirement definition
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* = `Scheduled`.
B11.FR.08	B11.FR.01 AND an *evseId* parameter is supplied AND ResetResponse has *status* = `Accepted`.	The Charging Station MAY send a NotifyEventRequest for *component* = "Connector", *variable* = "AvailabilityState", *actualValue* = "Unavailable" and *trigger* = "Delta" for the EVSE, and SHALL start a reset of the EVSE that is referred to by *evseId* parameter.
B11.FR.09	B11.FR.01 AND an *evseId* parameter is supplied AND Charging Station does not support resetting an individual EVSE	The Charging Station SHALL return a ResetResponse with *status* = `Rejected`
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

No.	Type	Description
1	**Name**	Reset - With Ongoing Transaction
2	**ID**	B12
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 paramater *evseId*, 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	**1.** The CSO requests the CSMS to reset the Charging Station or EVSE. **2.** The CSMS sends ResetRequest requesting the Charging Station to reset itself or EVSE. **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 *evseId* was supplied) that are Available to status *Unavailable*, waits until all transactions are finished and all TransactionEventRequest (eventType = Ended) messages are sent. **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 *evseId* was supplied) in progress, and sending a TransactionEventRequest (eventType = Ended) message. **4.** Only if no evseId was supplied the Charging Station reboots and returns to a state as just having been booted, B01 - Cold Boot Charging Station applies.
	Alternative scenario(s)	B11 - Reset Without Ongoing Transaction
5	**Prerequisite(s)**	A transaction is ongoing.
6	**Postcondition(s)**	**Successful postcondition:** The Charging Station was able to reset itself or EVSE. **Failure postcondition:** The Charging Station *not* was able to reset itself or EVSE.
7	**Error handling**	After having accepted the ResetRequest, TransactionEventRequest messages that cannot be delivered to the CSMS MUST be queued.
8	**Remark(s)**	n/a

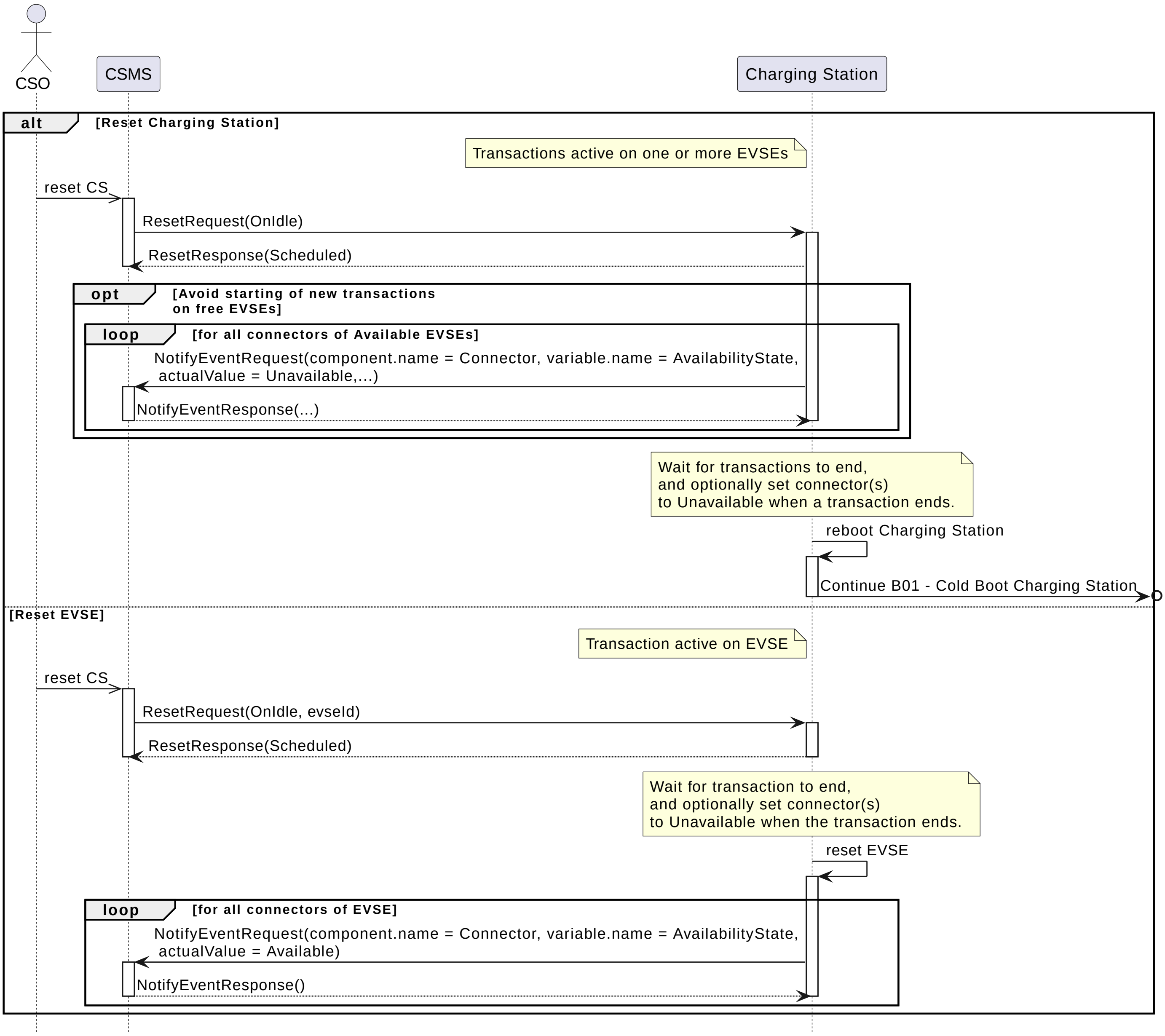

CSO
CSMS
Charging Station
alt [Reset Charging Station]
Transactions active on one or more EVSEs
reset CS
ResetRequest(OnIdle)
ResetResponse(Scheduled)
opt [Avoid starting of new transactions on free EVSEs]
loop [for all connectors of Available EVSEs]
NotifyEventRequest(component.name = Connector, variable.name = AvailabilityState, actualValue = Unavailable,...)
NotifyEventResponse(...)
Wait for transactions to end, and optionally set connector(s) to Unavailable when a transaction ends.
reboot Charging Station
Continue B01 - Cold Boot Charging Station
[Reset EVSE]
Transaction active on EVSE
reset CS
ResetRequest(OnIdle, evseId)
ResetResponse(Scheduled)
Wait for transaction to end, and optionally set connector(s) to Unavailable when the transaction ends.
reset EVSE
loop [for all connectors of EVSE]
NotifyEventRequest(component.name = Connector, variable.name = AvailabilityState, actualValue = Available)
NotifyEventResponse()

Figure 21a: Sequence Diagram: Reset OnIdle With Ongoing Transaction

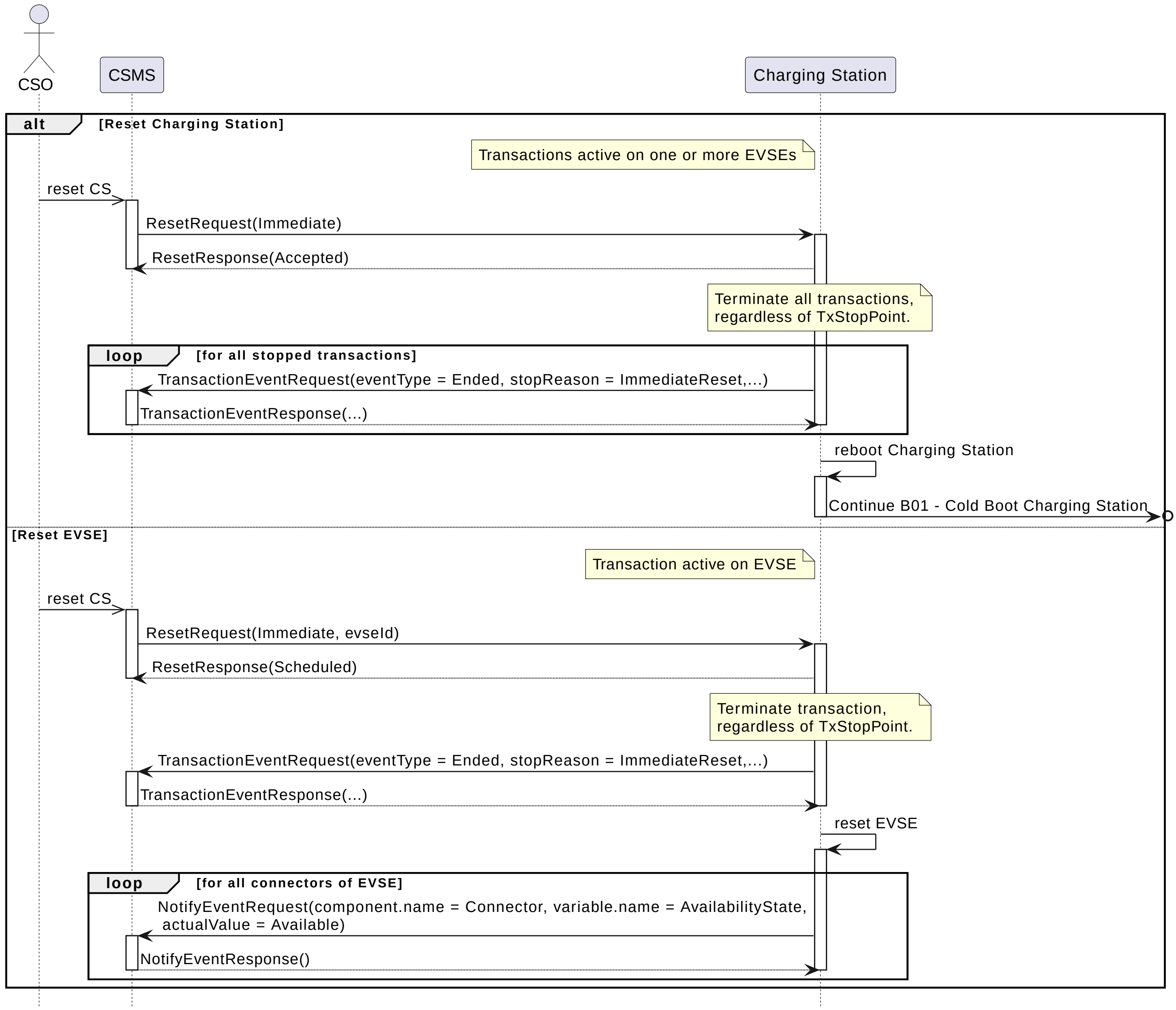

CSO
CSMS
Charging Station
alt
[Reset Charging Station]
Transactions active on one or more EVSEs
reset CS
ResetRequest(Immediate)
ResetResponse(Accepted)
Terminate all transactions, regardless of TxStopPoint.
loop
[for all stopped transactions]
TransactionEventRequest(eventType = Ended, stopReason = ImmediateReset,...)
TransactionEventResponse(...)
reboot Charging Station
Continue B01 - Cold Boot Charging Station
[Reset EVSE]
Transaction active on EVSE
reset CS
ResetRequest(Immediate, evseId)
ResetResponse(Scheduled)
Terminate transaction, regardless of TxStopPoint.
TransactionEventRequest(eventType = Ended, stopReason = ImmediateReset,...)
TransactionEventResponse(...)
reset EVSE
loop
[for all connectors of EVSE]
NotifyEventRequest(component.name = Connector, variable.name = AvailabilityState, actualValue = Available)
NotifyEventResponse()

Figure 21b: Sequence Diagram: Reset Immediate With Ongoing Transaction

B12 - Reset - With Ongoing Transaction - Requirements

Table 42. B12 - Requirements

ID	Precondition	Requirement definition
B12.FR.01	When the Charging Station receives a ResetRequest(OnIdle) AND a transaction is ongoing	The Charging Station SHALL respond with a ResetResponse(Scheduled), 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 ResetRequest(Immediate) AND a transaction is ongoing	The Charging Station SHALL respond with a ResetResponse(Accepted), to indicate whether the Charging Station will attempt to reset itself or EVSE.
B12.FR.03	If no *evseId* 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, e.g. as in E06 - Stop Transaction.
B12.FR.04	If no *evseId* 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 with *triggerReason* = `ResetCommand` and *transactionInfo.stoppedReason* = `ImmediateReset` for each terminated transaction before performing a reboot.

ID	Precondition	Requirement definition
B12.FR.05	If an Immediate Reset without *evseId* 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 *Inoperative* by the CSMS.	After a reboot of the Charging Station or resetting of EVSE, the EVSE(s) SHALL return to the state *Unavailable* as prior to the reboot of Charging Station or reset of EVSE. (Same as B11.FR.02)
B12.FR.07	If an *evseId* 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 reset, e.g. as in E06 - Stop Transaction.
B12.FR.08	If an *evseId* 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 with *triggerReason* = `ResetCommand` and *transactionInfo.stoppedReason* = `ImmediateReset` before resetting the EVSE.
B12.FR.09	B12.FR.01 AND an *evseId* parameter is supplied AND Charging Station does not support resetting an individual EVSE	The Charging Station SHALL return a ResetResponse with *status* = `Rejected` (Same as B11.FR.09)
B12.FR.10	B12.FR.02 AND Charging Station is at this moment not able to perform an Immediate reset for a reason other than the fact that a transaction is in progress	The Charging Station SHALL return a ResetResponse with *status* = `Rejected`

B13 - Reset - With Ongoing Transaction - Resuming Transaction

New in OCPP 2.1

No.	Type	Description
1	**Name**	Reset - With Ongoing Transaction - Resuming Transaction
2	**ID**	B13
3	**Objective(s)**	To enable the CSMS to request a Charging Station to reset itself or EVSE, while a transaction is ongoing, that is to be resumed after reset.
4	**Description**	This use case covers how the CSMS can request the Charging Station to reset itself or an EVSE by sending ResetRequest with *type* = `ImmediateAndResume`. (If ResetRequest contains an optional paramater *evseId*, 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 configuration variable `TxAllowEnergyTransferResumption` determines whether energy transfer is still allowed on the resumed transaction or not.
	Actors	Charging Station, CSMS, CSO

No.	Type	Description
	Scenario description	1. CSMS sends ResetRequest with *type* = `ImmediateAndResume` to Charging Station. 2. Charging Station responds with ResetResponse with *status* = `Accepted`. 3. Charging Station sends a TransactionEventRequest with *eventType* = `Updated` and *triggerReason* = `ResetCommand` to signal that a reset will be performed. 4. Charging Station remembers the charging state of ongoing transactions, stops energy transfer, and unlocks the connectors in case of detachable cables. 5. Charging Station will not end the ongoing transaction(s), but resume them after the reset. 6. Charging Station resets the entire Charging Station or EVSE a. Only when no *evseId* was supplied, the Charging Station reboots and returns to a state as just having been booted, B01 - Cold Boot Charging Station applies. 7. Charging Station locks the connectors again in case of detachable cables. 8. Charging Station resumes connectivity with the EVs and resumes the transactions in the charging state they had before the reset. a. If `TxAllowEnergyTransferResumption` = true, the energy transfer on the transaction will be resumed. b. If `TxAllowEnergyTransferResumption` = false, the energy transfer on the transaction will not be resumed. 9. Charging Station sends a TransactionEventRequest with *eventType* = `Updated` and *triggerReason* = `TxResumed` to signal that the transaction has been resumed.
	Alternative scenario(s)	B11 - Reset Without Ongoing Transaction B12 - Reset With Ongoing Transaction.
5	**Prerequisite(s)**	A transaction is ongoing.
6	**Postcondition(s)**	**Successful postcondition:** The Charging Station was able to reset itself or EVSE. The transaction continues and energy transfer only continues when `TxAllowEnergyTransferResumption` = true. **Failure postcondition:** - The Charging Station was not able to reset itself or its EVSE, - The Charging Station was not able to resume the transaction, - The EV was not able to resume the energy transfer.
7	**Error handling**	After having accepted the ResetRequest, any TransactionEventRequest messages that cannot be delivered to the CSMS MUST be queued.
8	**Remark(s)**	The EVSE of the Charging Station will have to resume the charging session towards the EV. Depending on the type of connector and protocol (Mode 3, ISO 15118, CHAdeMO, etc.) this will involve different actions, which are not described in this specification. Only when the configuration variable `TxResumptionTimeout` exists and has a value greater than zero, a ResetRequest with *type* = `ImmediateAndResume` is supported. After the reset all transactions are resumed, regardless of the value of `TxResumptionTimeout`. Charging Station may reject the reset request when it is unable to perform it, for example when it is busy with a firmware update process.

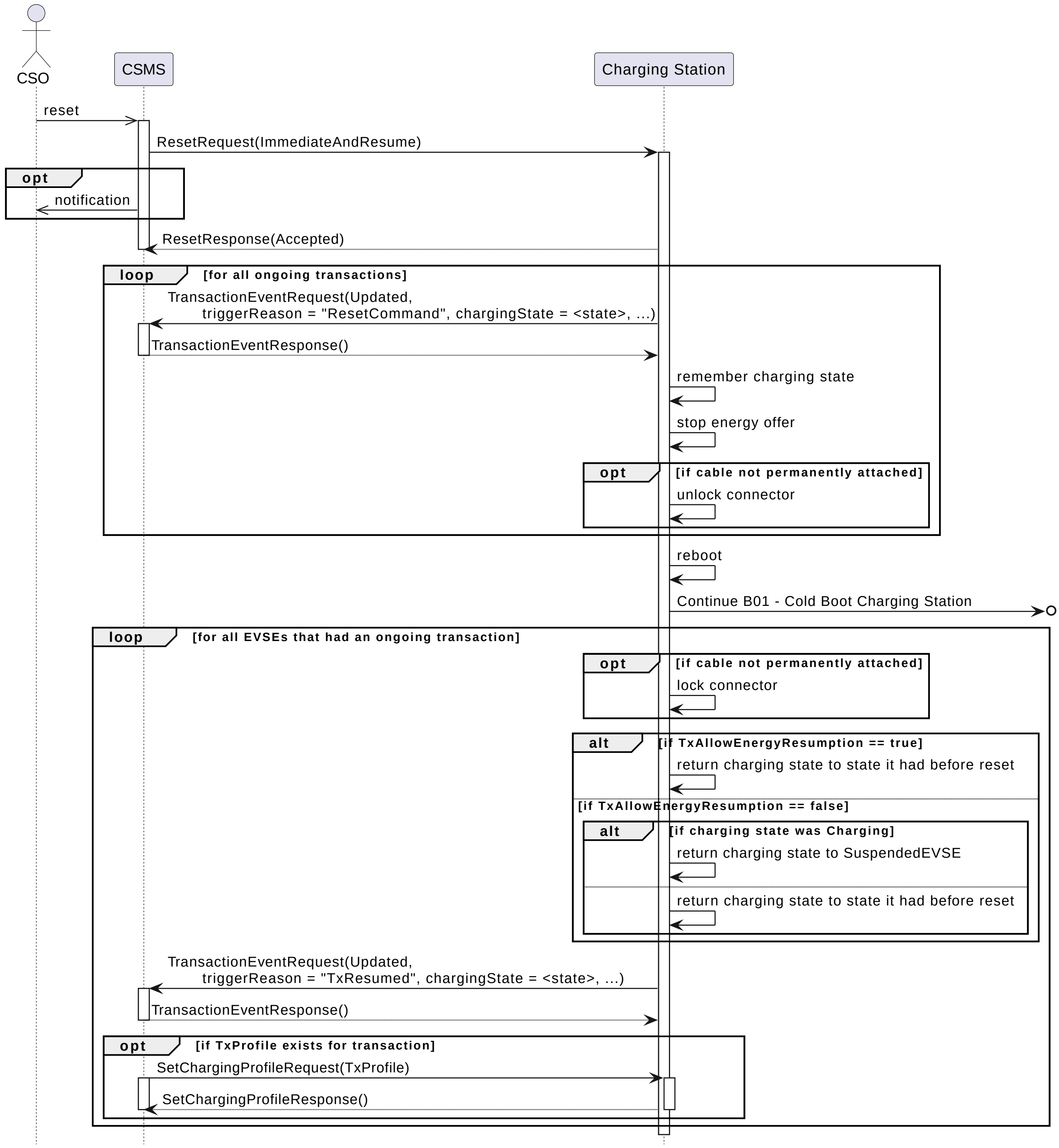

CSO
CSMS
Charging Station
reset
ResetRequest(ImmediateAndResume)
opt
notification
ResetResponse(Accepted)
loop [for all ongoing transactions]
TransactionEventRequest(Updated, triggerReason = "ResetCommand", chargingState = <state>, ...)
TransactionEventResponse()
remember charging state
stop energy offer
opt [if cable not permanently attached]
unlock connector
reboot
Continue B01 - Cold Boot Charging Station
loop [for all EVSEs that had an ongoing transaction]
opt [if cable not permanently attached]
lock connector
alt [if TxAllowEnergyResumption == true]
return charging state to state it had before reset
[if TxAllowEnergyResumption == false]
alt [if charging state was Charging]
return charging state to SuspendedEVSE
return charging state to state it had before reset
TransactionEventRequest(Updated, triggerReason = "TxResumed", chargingState = <state>, ...)
TransactionEventResponse()
opt [if TxProfile exists for transaction]
SetChargingProfileRequest(TxProfile)
SetChargingProfileResponse()

Figure 21. Sequence Diagram: Reset With Ongoing Transaction

B13- Reset - With Ongoing Transaction - Without Termination - Requirements

Table 43. B13 - Requirements

ID	Precondition	Requirement definition
B13.FR.01	When Charging Station receives a ResetRequest with *type* = `ImmediateAndResume` AND `TxResumptionTimeout` == 0 or non-existent	Charging Station SHALL respond with ResetResponse with *status* = `Rejected`. (When transaction resumption is not supported by this Charging Station)
B13.FR.02	When Charging Station receives a ResetRequest with *type* = `ImmediateAndResume` AND No transactions are ongoing on Charging Station	Charging Station SHALL respond with ResetResponse with *status* = `Accepted` and act according to use case B11 - Reset - Without Ongoing Transactions.

ID	Precondition	Requirement definition
B13.FR.03	When Charging Station receives a ResetRequest with *type* = `ImmediateAndResume` AND Transactions are ongoing on Charging Station	Charging Station SHALL respond with ResetResponse with *status* = `Accepted`
B13.FR.04	B13.FR.03	Charging Station SHALL send TransactionEventRequest with *eventType* = `Updated` and *triggerReason* = `ResetCommand` and any other required parameters for each ongoing transaction.
Reset of EVSE		
B13.FR.10	B13.FR.04 AND parameter *evseId* had a value	Charging Station SHALL stop energy transfer for the transaction on *evseId*.
B13.FR.11	B13.FR.10	Charging Station SHALL reset the EVSE matching *evseId* without rebooting the entire Charging Station.
B13.FR.12	B13.FR.11 AND After reset is completed	Charging Station SHALL resume the transaction in the state it had before the reset, even when resetting took longer than `TxResumptionTimeout`.
B13.FR.13	B13.FR.12 AND `TxAllowEnergyTransferResumption` = true	Charging Station SHALL allow energy transfer in the resumed transaction.
B13.FR.14	B13.FR.12 AND `TxAllowEnergyTransferResumption` = false AND *transactionInfo.chargingState* = `Charging`	Charging Station SHALL resume the transaction in the *chargingState* = `SuspendedEVSE` (i.e. no energy transfer)
Reset of Charging Station		
B13.FR.20	B13.FR.04 AND parameter *evseId* is not present	Charging Station SHALL stop energy transfer for all transactions and unlock the connectors in case of detachable cables.
B13.FR.21	B13.FR.20	Charging Station SHALL reboot the entire Charging Station.
B13.FR.22	B13.FR.21 AND After reboot is completed	Charging Station SHALL lock the connectors that had ongoing transaction in case of detachable cables.
B13.FR.23	B13.FR.22	Charging Station SHALL resume each transaction in the state it had before the reset, even when resetting took longer than `TxResumptionTimeout`.
B13.FR.24	B13.FR.23 AND `TxAllowEnergyTransferResumption` = true	Charging Station SHALL allow energy transfer in resumed transactions.
B13.FR.25	B13.FR.23 AND `TxAllowEnergyTransferResumption` = false AND *transactionInfo.chargingState* of a transaction is `Charging`	Charging Station SHALL resume such a transaction in the *chargingState* `SuspendedEVSE` (i.e. no energy transfer)
Signal resumption		
B13.FR.30	B13.FR.12 OR B13.FR.23	Charging Station SHALL send a TransactionEventRequest with *eventType* = `Updated` and *triggerReason* = `TxResumed` for each resumed transaction.
B13.FR.31	B13.FR.30 AND CSMS has a charging profile of *chargingProfilePurpose* = `TxProfile` for transactions reported with *triggerReason* = `TxResumed` AND SmartChargingCtrlr.ChargingProfilePersistence for *instance* = "TxProfile" = false or absent	CSMS SHALL send the SetChargingProfileRequest for *chargingProfilePurpose* = `TxProfile` for these transactions again.

C. Authorization

Chapter 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

Updated in OCPP 2.1

The field *additionalInfo* of IdTokenType can be used to send extra information about the *idToken* to the CSMS. This can be used in specific situations. For example, when known, the license plate of the vehicle can be added as *additionalInfo*, which can then be passed on to a parking service, or in the case of ad hoc payment with a credit card, the *additionalInfo* can contain information about the credit card authorization.

When comparing two *idTokens*, the *additionalInfo* is disregarded. This also means that the *additionalInfo* should not be stored in the Local Authorization Cache as part of IdTokenType.

The meaning of an *additionalInfo* field is determined by its *type* field. This is a free format string. The supported values need to be agreed upon between Charging Station and CSMS. A limited set of values for *additionalInfo.type*, related to ad hoc payment support, is defined in [OCPP2.1-PART2-APPENDIX].

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. IDTokenTypes 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 IdTokenTypes 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.

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 in *cacheExpiryDateTime*. 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.

1.6. Relationship between authorization and transaction

This section is informative.

The purpose of authorization is twofold. It ensures in the first place, that energy is only offered to a known user (represented by the *idToken*), which is essential for billing. In the second place, it ensures that only the user who was authorized in the first place (or a member of the same group of users) is allowed to unplug the cable. This is an important safeguard against cable theft in situations where the charging station does not have a fixed cable and the user brings its own charging cable.

Authorization and the duration of the authorization period are not strictly tied to a transaction: it is possible to have transactions without explicit authorization, e.g. in the case of a charging station that can be started with a push button. In that case one could say that there is a permanent authorization for anyone to charge.

The start of the authorization period:

- can take place before a transaction is started (e.g. when a cable is not yet connected), or
- can cause a transaction to be started (e.g. when authorization is defined as the start of a transaction by setting TxStartPoint = Authorized), or
- can happen after a transaction has already started (e.g. when connection of the cable is defined as the start of a transaction by setting TxStartPoint = EVConnected).

(See chapter E.1.1 "Flexible transaction start/stop" for a description of transaction start and stop points.)

In any case, authorization (or authorization period) ends when the same *idToken* is presented again for authorization, or when the transaction ends. This means that ending of the authorization period:

- can happen during a transaction without ending the transaction (e.g. when *idToken* is presented again, but TxStopPoint = EVConnected), or
- can cause the transaction to end (e.g. when *idToken* is presented again and TxStopPoint = Authorized), or
- can be caused by the end of the transaction (e.g. when *idToken* is not presented for authorization, but the cable is disconnected and TxStopPoint = EVConnected), or
- can be caused by cable plug-out if no transaction was started.

A Charging Station defines when authorization starts (i.e. upon receiving the AuthorizeResponse, or when authorizing locally via authorization cache or local authorization list) and when authorization ends (i.e. when *idToken* is presented a second time, or when the transaction ends). Charging Station notifies CSMS about this, as follows:

- If authorization occurs before start of the transaction, Charging Station tells CSMS that authorization has taken place, by including the *idToken* in the first TransactionEventRequest of the transaction.
- If authorization occurs within a transaction or at the start of a transaction, Charging Station reports this by including the *idToken* in TransactionEventRequest together with a *triggerReason* = Authorized.
- The end of authorization is reported in a TransactionEventRequest with a *triggerReason* = StopAuthorized or Deauthorized, or by reporting the end of the transaction.

Chapter 2. Use cases & Requirements

2.1. Authorization options

C01 - EV Driver Authorization using RFID

No.	Type	Description
1	**Name**	EV Driver Authorization using RFID
2	**ID**	C01
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	**1.** The EV Driver wants to start or stop charging the EV and presents an RFID card. **2.** The Charging Station sends AuthorizeRequest to the CSMS to request authorization. **3.** Upon receipt of AuthorizeRequest, the CSMS responds with AuthorizeResponse. This response message indicates whether or not the IdToken is accepted by the CSMS.
	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)**	**Successful postcondition:** The EV Driver is authorized and can start *or* stop charging. **Failure postcondition:** If the authorize message is *Invalid*, *Blocked*, *Expired* or *Unknown*, the EV Driver can *not* start or stop charging, except in the case where the EV Driver presents the same token used to start the transaction.
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 *idToken* is valid for charging and the Charging Station has 3 EVSEs, what is the content of *idTokenInfo*, when *idToken* is allowed to charge: . at all EVES: *idTokenInfo.status* = `Accepted`. . at EVSE 1: *idTokenInfo.status* = `Accepted`, *idTokenInfo.evseId* = [1]. . at EVSE 1 + 2: *idTokenInfo.status* = `Accepted`, *idTokenInfo.evseId* = [1, 2]. . at none of the EVSEs: _idTokenInfo.status=`NotAtThisLocation`.

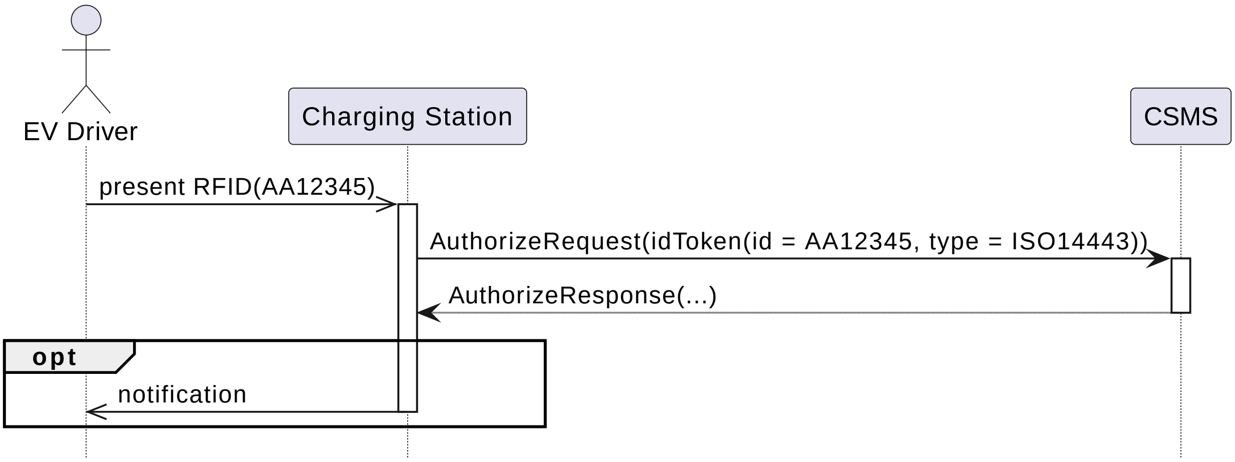


Figure 22. Sequence Diagram: EV Driver Authorization

C01 - EV Driver Authorization using RFID - Requirements

Table 44. 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 *language*.	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 *language* AND if the field: language2 is set AND the Charging Station contains messages in that *language*	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.	

ID	Precondition	Requirement definition	Note
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*.	
C01.FR.19	*idToken* is allowed for any EVSE of the Charging Station	The CSMS SHALL send an AuthorizeResponse in which *idTokenInfo* has an empty (or absent) *evseId* list.	This will be the most common case. Even though the *idToken* might be allowed on any EVSE, the *idTokenInfo.status* still needs to be `Accepted` before charging is allowed.
C01.FR.20	*idToken* is allowed for a subset of EVSEs of the Charging Station	The CSMS SHALL send an AuthorizeResponse in which *IdTokenInfo* has an *evseId* list with the allowed EVSEs.	Note the difference between validity of an *idToken* and the fact whether this (type of) token is allowed on an EVSE. The *idTokenInfo.status* 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	*idToken* is not allowed for any EVSE of the Charging Station	The CSMS SHALL send an AuthorizeResponse in which *idTokenInfo.status* is `NotAtThisLocation` and *evseId* list is empty (or absent).	Status `NotAtThisLocation` needed in order to differentiate with the situation in which *idToken* is allowed on all EVSEs.
C01.FR.23	When a transaction is still active, that had been authorized earlier by an idToken, but which is now no longer authorized for charging AND a new idToken is presented to the Charging Station for authorization, that **differs** from the inital idToken	The Charging Station SHOULD NOT allow the authorization of a different idToken.	Multiple *idTokens* 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 idToken, but which is now no longer authorized for charging AND Charging Stations sends an AuthorizeRequest for a new idToken, that **differs** from the inital idToken of the transaction	The CSMS is RECOMMENDED to respond with an AuthorizeResponse with *idTokenInfo.status* = `NotAtThisTime` for this idToken.	If a second authorization is done by Charging Station then CSMS can reject the *idToken*.
C01.FR.25		Two IdTokenType elements are considered to be equal when they have the same value for the fields *idToken.idToken* and *idToken.type*	*additionalInfo* is not taken into account when comparing. See C01.FR.02, C01.FR.03, C01.FR.05 for *idToken* requirements where *idTokens* are compared.

ID	Precondition	Requirement definition	Note
C01.FR.26	When an *idToken* has been authorized and the EV Driver does not plug in the charging cable before the timeout set by the Configuration Variable: `EVConnectionTimeOut`	Charging Station SHALL end the authorization of *idToken*	See also E03.FR.05 and F02.FR.07/08 for additional behavior in case a transaction had already been started.

C02 - Authorization using a start button

No.	Type	Description
1	**Name**	Authorization using a start button
2	**ID**	C02
3	**Objective(s)**	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 start charging, it is not needed to send an AuthorizeRequest. In the TransactionEventRequest (eventType = Started), IdTokenType information needs to be given, which the CSMS then cannot reject.
	Actors	EV Driver, Charging Station, CSMS
	Scenario description	**1.** The EV Driver plugs in the charging cable between EV and Charging Station. **2.** The Charging Station sends a NotifyEventRequest with component.name *Connector*, variable.name *AvailabilityState*, actualValue *Occupied* and TransactionEventRequest (eventType = Started) 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 (eventType = Updated) message with *idToken.type* = `NoAuthorization` to the CSMS to notify the CSMS of the charging that has started. **6.** Upon receipt of TransactionEventRequest (eventType = Updated), the CSMS responds with TransactionEventResponse with: IdTokenInfo.status set to *Accepted*
	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.
7	**Error Handling**	n/a
8	**Remarks**	The start button might also be a mechanical key or something similar. Note that the start button can even be omitted if the Charging Station is configured to start charging upon cable connection. The scenario description and sequence diagram above are based on the Configuration Variable for start transaction being configured as follows: `TxStartPoint`: 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 send. For more details see the use case: E01 - Start Transaction options.

C02 - Authorization using a start button - Requirements

Table 45. C02 - 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.

ID.	Precondition	Requirement definition
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

Updated in OCPP 2.1

No.	Type	Description
1	**Name**	Authorization using credit card
2	**ID**	C03
3	**Objective(s)**	Make it possible to start a transaction using a credit card.
NOTE	This use case has been moved to section C24 - Ad hoc payment via stand-alone payment terminal as a new use case.	

C04 - Authorization using PIN-code

This is an informative use case. Its purpose is to demonstrate the use of the *idToken.type* `KeyCode`. Another use of `KeyCode` might be, for example, a licence plate number.

No.	Type	Description
1	**Name**	Authorization using PIN-code
2	**ID**	C04
3	**Objective(s)**	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
	Scenario description	**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 *idToken.type* 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.
	Alternative scenario(s)	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.
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.

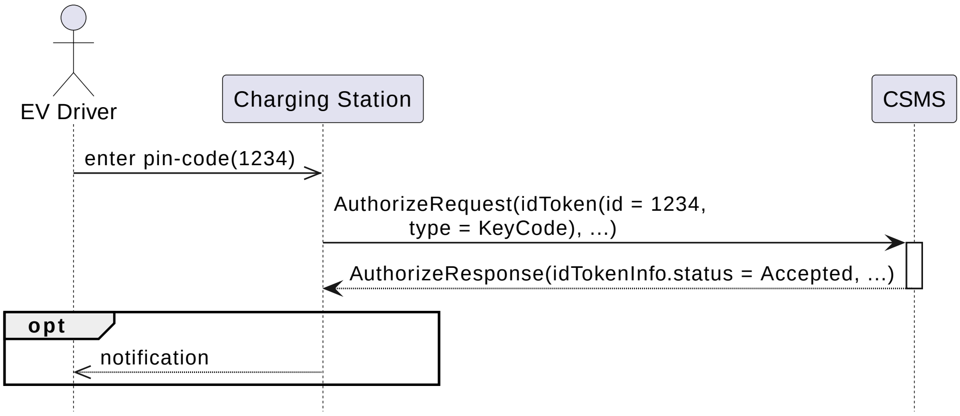

EV Driver
Charging Station
CSMS
enter pin-code(1234)
AuthorizeRequest(idToken(id = 1234,
type = KeyCode), ...)
AuthorizeResponse(idTokenInfo.status = Accepted, ...)
opt
notification

Figure 23. Sequence Diagram: Authorization using PIN-code

C04 - Authorization using PIN-code - Requirements

Table 46. C04 - 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 status = Invalid.
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.

ID.	Precondition	Requirement definition
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

No.	Type	Description
1	**Name**	Authorization for CSMS initiated transactions
2	**ID**	C05
3	**Objective(s)**	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	**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
7	**Error Handling**	n/a
8	**Remarks**	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. 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. The scenario description and sequence diagram above are based on the Configuration Variable for start transaction being configured as follows: `TxStartPoint`: 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 send. For more details see the use case: E01 - Start Transaction options. This use case assumes that the configuration variable AuthorizeRemoteStart is *false*. See use cases F01 and F02 for requirements with AuthorizeRemoteStart. Other idToken *types* can also be used to remote start charging, such an `eMAID` of the user that is provided by the app.

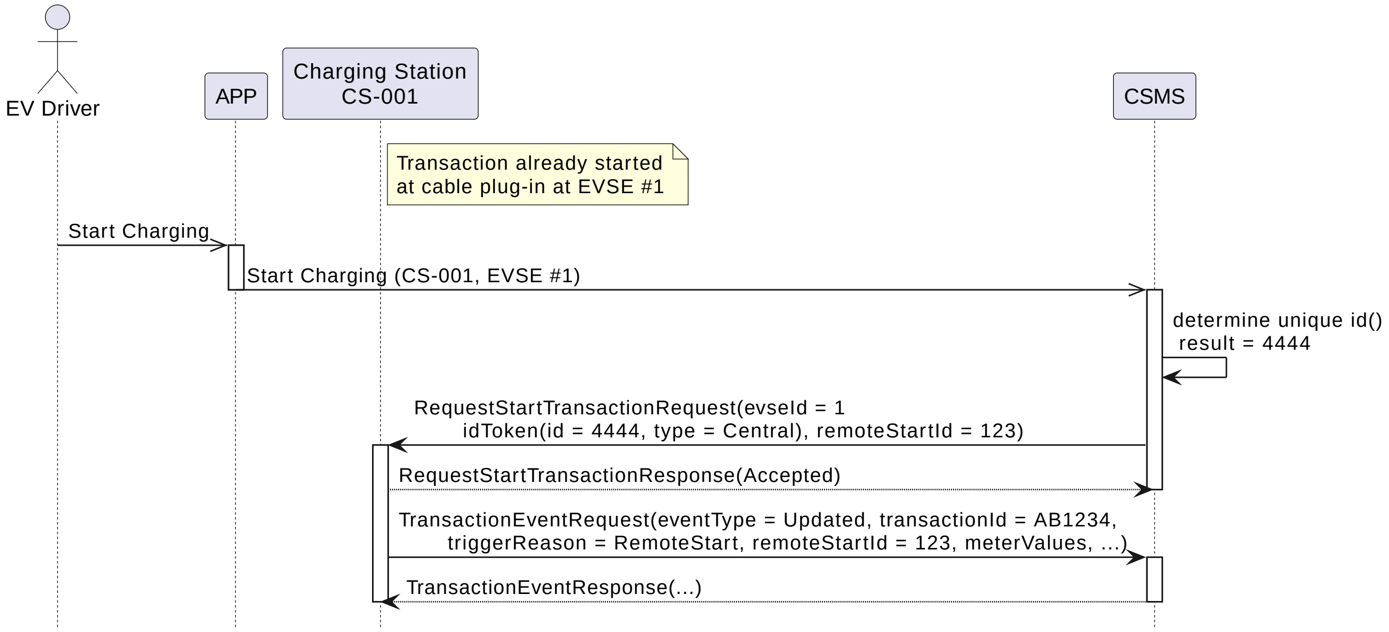

EV Driver
APP
Charging Station
CS-001
CSMS
Transaction already started
at cable plug-in at EVSE #1
Start Charging
Start Charging (CS-001, EVSE #1)
determine unique id()
result = 4444
RequestStartTransactionRequest(evseId = 1
idToken(id = 4444, type = Central), remoteStartId = 123)
RequestStartTransactionResponse(Accepted)
TransactionEventRequest(eventType = Updated, transactionId = AB1234,
triggerReason = RemoteStart, remoteStartId = 123, meterValues, ...)
TransactionEventResponse(...)

Figure 24. Sequence Diagram: Authorization for CSMS initiated transactions

C05 - Authorization for CSMS initiated transactions Requirements

Table 47. C05 - 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. (Same as C03.FR.02)
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 idToken *type* `Local`.

No.	Type	Description
1	**Name**	Authorization using local id type
2	**ID**	C06
3	**Objective(s)**	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
	Scenario description	**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.
	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 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.
7	**Error Handling**	n/a
8	**Remarks**	This use case uses an Parking Ticket as example, but this is not a requirement. The communication between the Payment Terminal and the CSMS is outside of scope of OCPP. The scenario description and sequence diagram above are based on the Configuration Variable for start & stop transaction being configured as follows: `TxStartPoint`: EVConnected `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 send. For more details see the use cases: E01 - Start Transaction options and E06 - Stop Transaction options.

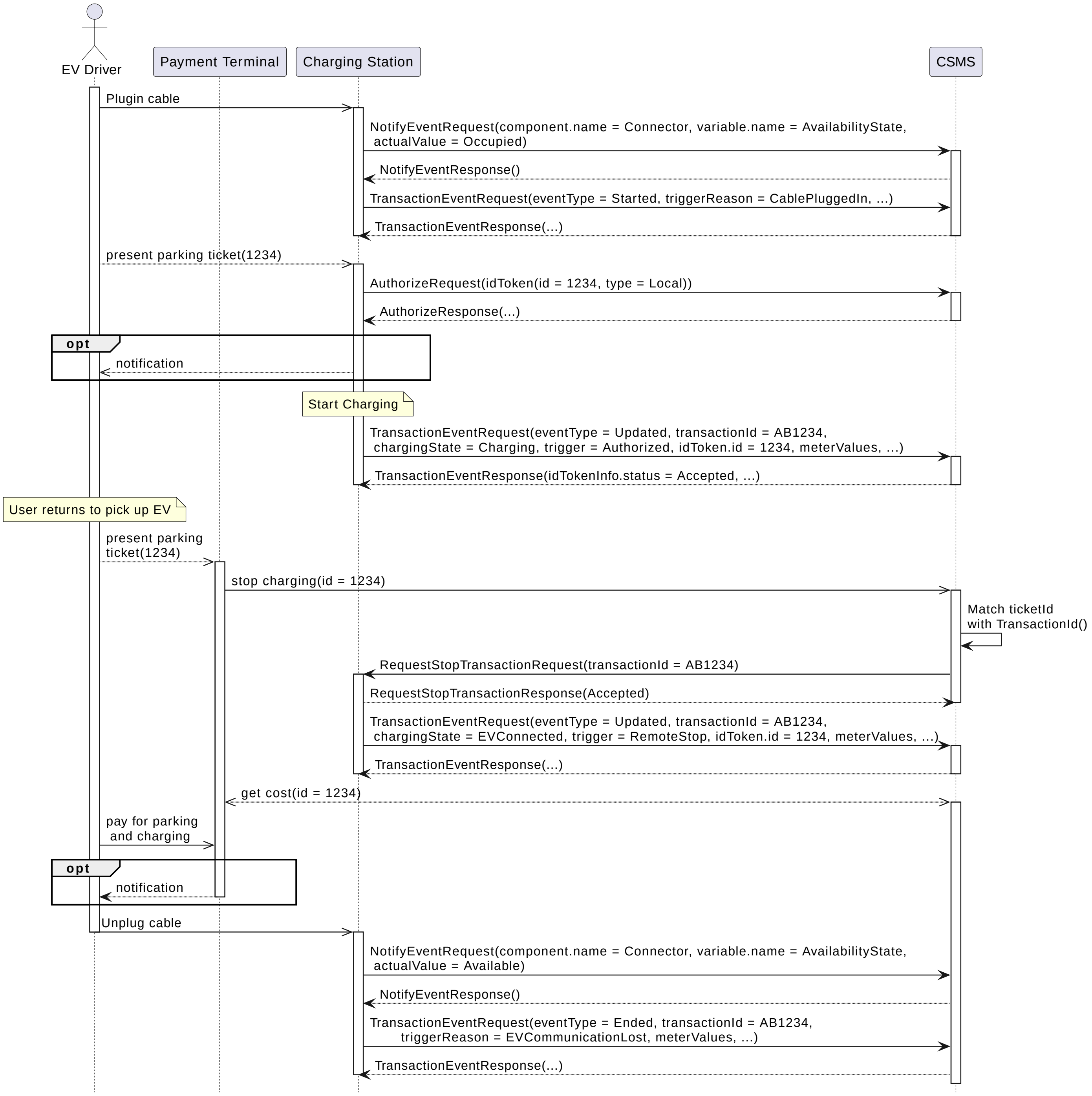

EV Driver
Payment Terminal
Charging Station
CSMS
Plugin cable
NotifyEventRequest(component.name = Connector, variable.name = AvailabilityState, actualValue = Occupied)
NotifyEventResponse()
TransactionEventRequest(eventType = Started, triggerReason = CablePluggedIn, ...)
TransactionEventResponse(...)
present parking ticket(1234)
AuthorizeRequest(idToken(id = 1234, type = Local))
AuthorizeResponse(...)
opt
notification
Start Charging
TransactionEventRequest(eventType = Updated, transactionId = AB1234, chargingState = Charging, trigger = Authorized, idToken.id = 1234, meterValues, ...)
TransactionEventResponse(idTokenInfo.status = Accepted, ...)
User returns to pick up EV
present parking ticket(1234)
stop charging(id = 1234)
Match ticketId with TransactionId()
RequestStopTransactionRequest(transactionId = AB1234)
RequestStopTransactionResponse(Accepted)
TransactionEventRequest(eventType = Updated, transactionId = AB1234, chargingState = EVConnected, trigger = RemoteStop, idToken.id = 1234, meterValues, ...)
TransactionEventResponse(...)
get cost(id = 1234)
pay for parking and charging
opt
notification
Unplug cable
NotifyEventRequest(component.name = Connector, variable.name = AvailabilityState, actualValue = Available)
NotifyEventResponse()
TransactionEventRequest(eventType = Ended, transactionId = AB1234, triggerReason = EVCommunicationLost, meterValues, ...)
TransactionEventResponse(...)

Figure 25. Sequence Diagram: Authorization using local id type

C06 - Authorization using local id type - Requirements

Table 48. C06 - Requirements

ID	Precondition	Requirement definition
C06.FR.01		The Charging Station SHALL only offer energy after authorization. (Same as C01.FR.01)
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. (Same as C01.FR.04)
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

Updated in OCPP 2.1

No.	Type	Description
1	**Name**	Authorization using Contract Certificates
2	**ID**	C07
	Reference	ISO15118-1 D2
3	**Objective(s)**	See ISO15118-1, use case Objective D2, page 26.
4	**Description**	See ISO15118-1, use case Description D2 (first bullet), page 26.
	Actors	EV, Charging Station, CSMS
	Scenario description	**15118**: See ISO15118-1, use case Description D2, Scenario Description, first 2 bullets, page 26. **OCPP**: **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.
	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 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.
7	**Error handling**	
8	**Remark(s)**	In edition 1 of 15118, the message timeout of the PaymentDetailsReq/Res message is 5 seconds. In case certificate verification cannot be completed in that time it is possible to complete this during the AuthorizationReq/Res, which can be extended up to 60 seconds. When the Charging Station is offline, it is recommended to omit the payment option for ISO 15118 contract certificates from the ServiceDiscoveryRes and revert to External Identification Means (use case C08), because certificate status cannot be checked.

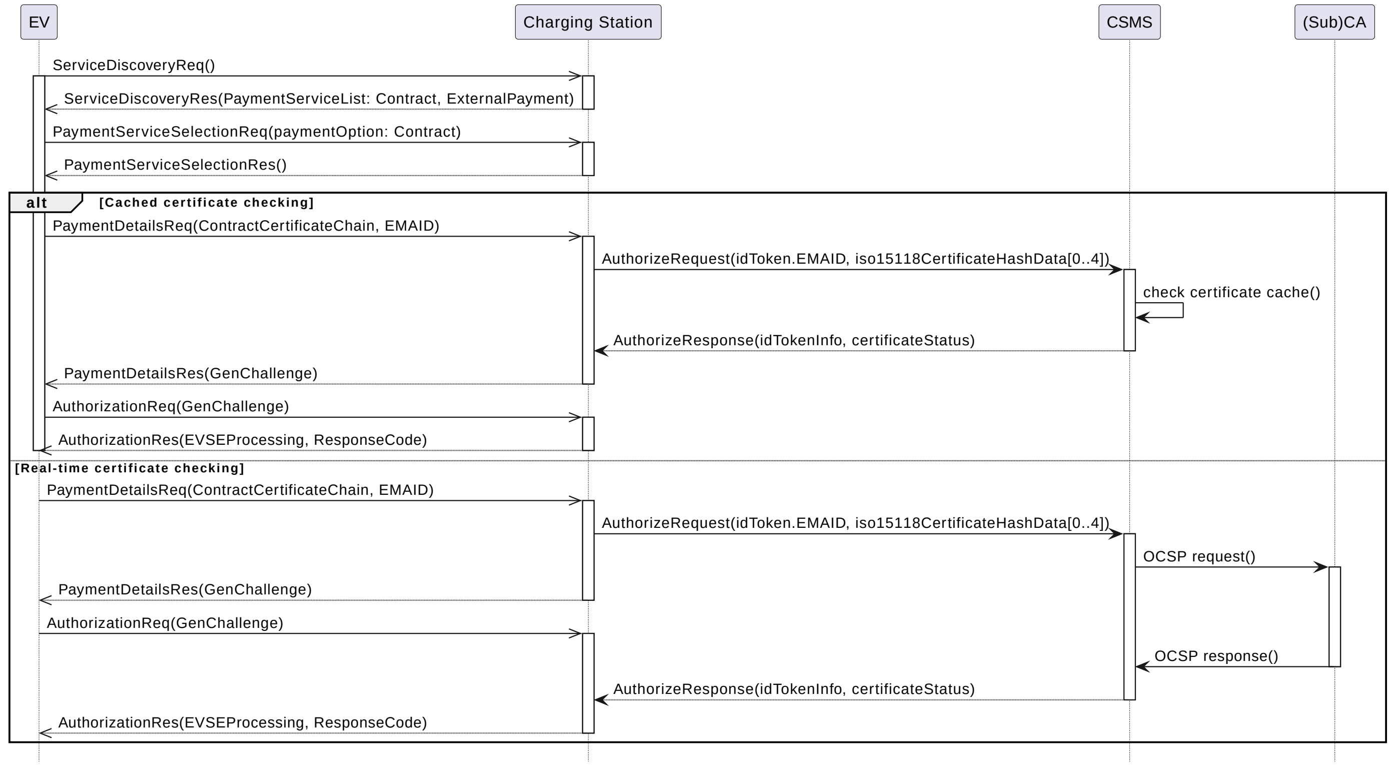

EV
Charging Station
CSMS
(Sub)CA
ServiceDiscoveryReq()
ServiceDiscoveryRes(PaymentServiceList: Contract, ExternalPayment)
PaymentServiceSelectionReq(paymentOption: Contract)
PaymentServiceSelectionRes()
alt
[Cached certificate checking]
PaymentDetailsReq(ContractCertificateChain, EMAID)
AuthorizeRequest(idToken.EMAID, iso15118CertificateHashData[0..4])
check certificate cache()
AuthorizeResponse(idTokenInfo, certificateStatus)
PaymentDetailsRes(GenChallenge)
AuthorizationReq(GenChallenge)
AuthorizationRes(EVSEProcessing, ResponseCode)
[Real-time certificate checking]
PaymentDetailsReq(ContractCertificateChain, EMAID)
AuthorizeRequest(idToken.EMAID, iso15118CertificateHashData[0..4])
OCSP request()
PaymentDetailsRes(GenChallenge)
AuthorizationReq(GenChallenge)
OCSP response()
AuthorizeResponse(idTokenInfo, certificateStatus)
AuthorizationRes(EVSEProcessing, ResponseCode)

Figure 26. Authorization using Contract Certificates with ISO 15118-2

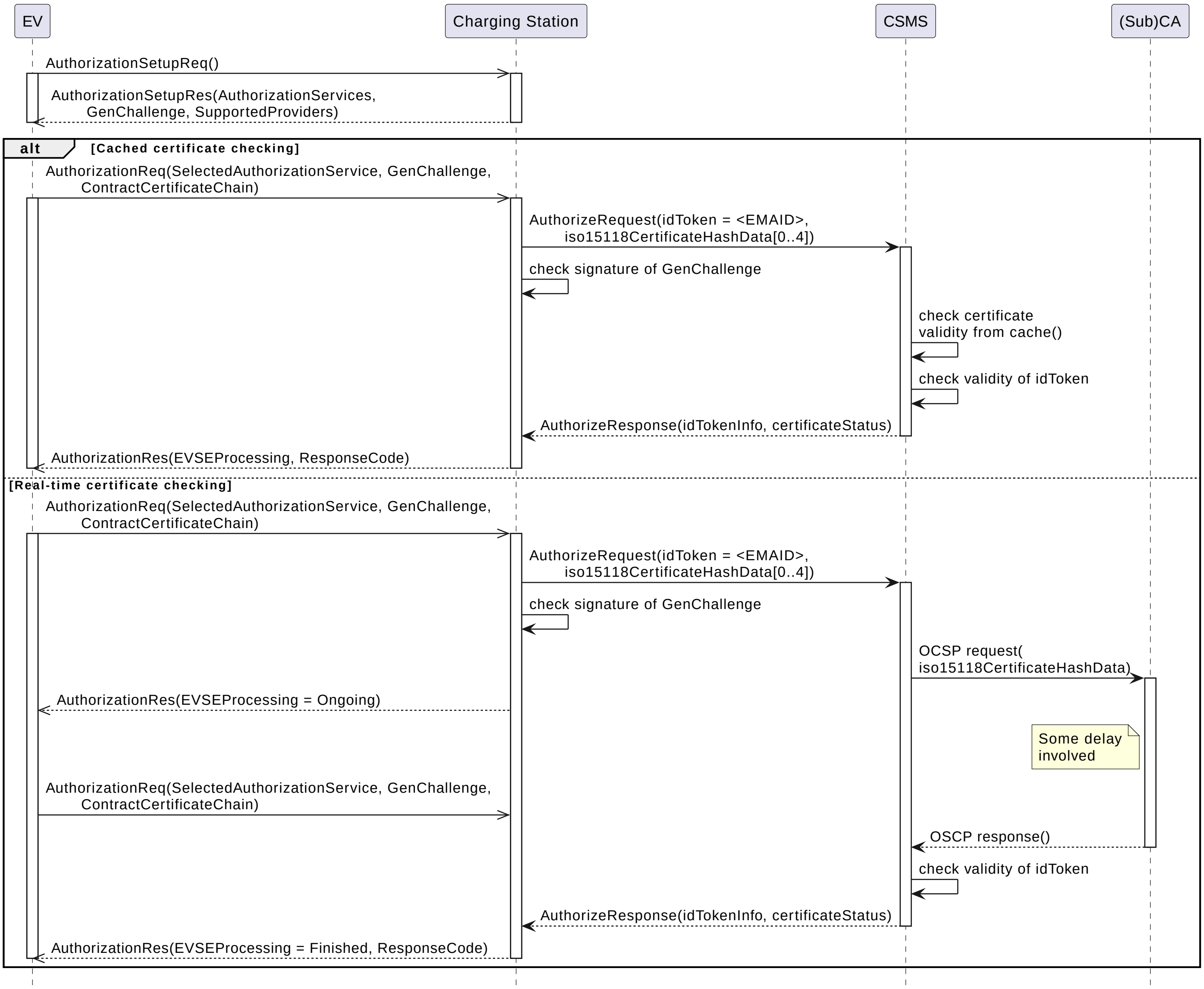

EV
Charging Station
CSMS
(Sub)CA
AuthorizationSetupReq()
AuthorizationSetupRes(AuthorizationServices, GenChallenge, SupportedProviders)
alt
[Cached certificate checking]
AuthorizationReq(SelectedAuthorizationService, GenChallenge, ContractCertificateChain)
AuthorizeRequest(idToken = <EMAID>, iso15118CertificateHashData[0..4])
check signature of GenChallenge
check certificate validity from cache()
check validity of idToken
AuthorizeResponse(idTokenInfo, certificateStatus)
AuthorizationRes(EVSEProcessing, ResponseCode)
[Real-time certificate checking]
AuthorizationReq(SelectedAuthorizationService, GenChallenge, ContractCertificateChain)
AuthorizeRequest(idToken = <EMAID>, iso15118CertificateHashData[0..4])
check signature of GenChallenge
OCSP request(iso15118CertificateHashData)
AuthorizationRes(EVSEProcessing = Ongoing)
Some delay involved
AuthorizationReq(SelectedAuthorizationService, GenChallenge, ContractCertificateChain)
OSCP response()
check validity of idToken
AuthorizeResponse(idTokenInfo, certificateStatus)
AuthorizationRes(EVSEProcessing = Finished, ResponseCode)

Figure 27. Authorization using contract certificates with ISO 15118-20

C07 - Authorization using Contract Certificates - Requirements

Table 49. C07 - Requirements

ID	Precondition	Requirement definition	Note
C07.FR.01	When Charging Station is online	The Charging Station SHALL send an AuthorizeRequest to the CSMS for validation.	
C07.FR.02	C07.FR.01	The AuthorizeRequest 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 AuthorizeRequest.	It SHALL respond with an AuthorizeResponse 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.	
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 `CentralContractValidationAllowed` is *true*	The Charging Station SHALL pass the contract certificate chain to the CSMS in *certificate* attribute (in PEM format) of AuthorizeRequest for validation by CSMS.	
C07.FR.07	When Charging Station is offline AND `ContractValidationOffline` is *false*	The Charging Station SHALL NOT allow charging.	
C07.FR.08	When Charging Station is offline AND `ContractValidationOffline` is *true*	The Charging Station SHALL try to validate the contract certificate locally.	
C07.FR.09	C07.FR.08 AND Contract certificate is valid AND `LocalAuthorizeOffline` is *true*	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 `OfflineTxForUnknownIdEnabled` = *true*	The Charging Station SHALL allow charging according to use case C15 - Offline Authorization of unknown Id.	
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`).

ID	Precondition	Requirement definition	Note
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 cannot be verified, then status of *idToken* is reported as invalid.

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

Updated in OCPP 2.1

(EIM)

No.	Type	Description
1	**Name**	Authorization at EVSE using ISO 15118 External Identification Means (EIM)
2	**ID**	C08 / 15118-1 D4
	Reference	ISO15118-1 D4
3	**Objective(s)**	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	EV, Charging Station, CSMS
	Scenario description	**15118** See ISO15118-1, use case Description (Scenarion Description) D4, page 28. **OCPP** **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.
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.

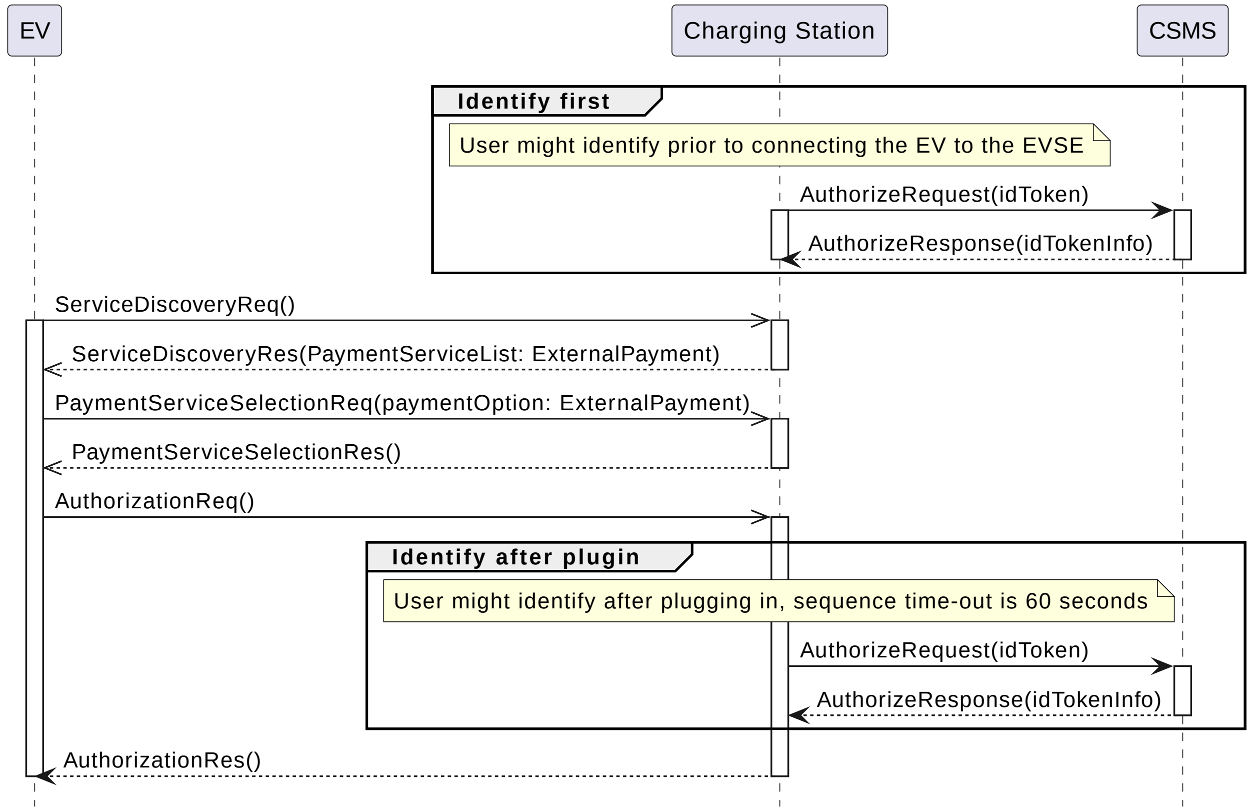

EV
Charging Station
CSMS
Identify first
User might identify prior to connecting the EV to the EVSE
AuthorizeRequest(idToken)
AuthorizeResponse(idTokenInfo)
ServiceDiscoveryReq()
ServiceDiscoveryRes(PaymentServiceList: ExternalPayment)
PaymentServiceSelectionReq(paymentOption: ExternalPayment)
PaymentServiceSelectionRes()
AuthorizationReq()
Identify after plugin
User might identify after plugging in, sequence time-out is 60 seconds
AuthorizeRequest(idToken)
AuthorizeResponse(idTokenInfo)
AuthorizationRes()

Figure 28. Sequence Diagram: Authorization using external identification means with ISO 15118-2

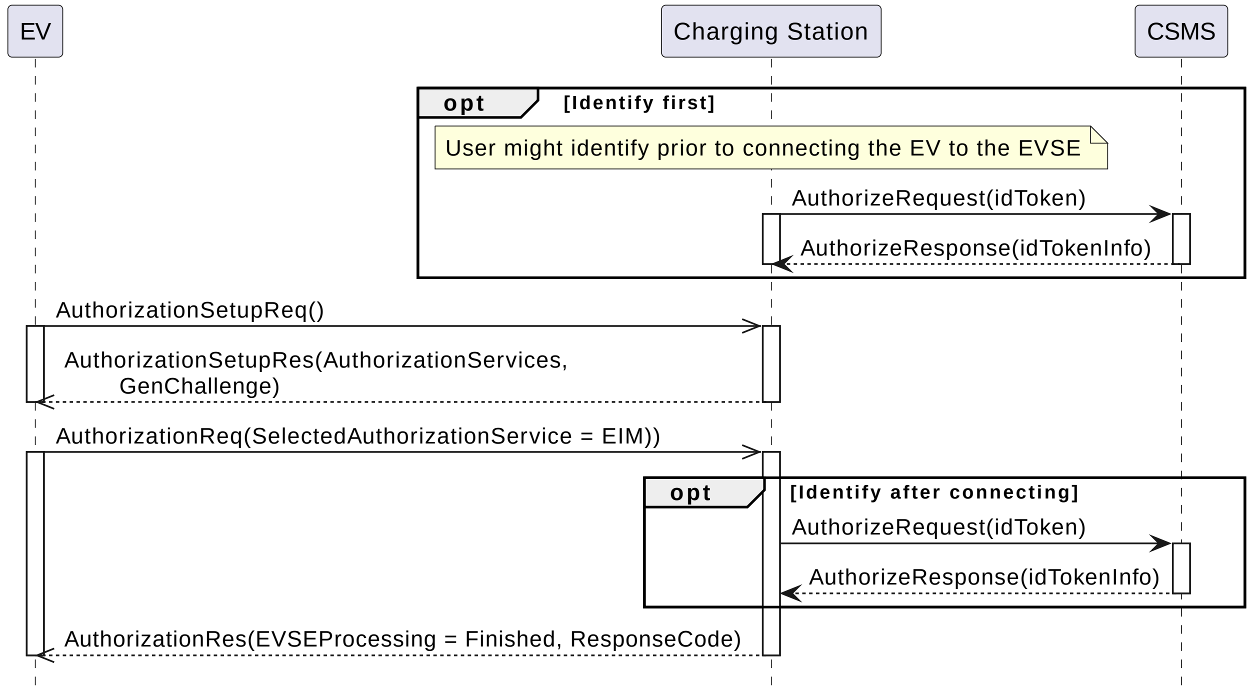

EV
Charging Station
CSMS
opt
[Identify first]
User might identify prior to connecting the EV to the EVSE
AuthorizeRequest(idToken)
AuthorizeResponse(idTokenInfo)
AuthorizationSetupReq()
AuthorizationSetupRes(AuthorizationServices, GenChallenge)
AuthorizationReq(SelectedAuthorizationService = EIM))
opt
[Identify after connecting]
AuthorizeRequest(idToken)
AuthorizeResponse(idTokenInfo)
AuthorizationRes(EVSEProcessing = Finished, ResponseCode)

Figure 29. Sequence Diagram: Authorization using external identification means with ISO 15118-20

Source: ISO15118-1

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

Table 50. C08 - Requirements

ID	Precondition	Requirement definition
C08.FR.01		The Charging Station SHALL send the identification to the CSMS for validation.

ID	Precondition	Requirement definition
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

No.	Type	Description
1	**Name**	Authorization by GroupId
2	**ID**	C09
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	**1.** EV Driver 1 presents an IdToken. **2.** The Charging Station sends AuthorizeRequest to the CSMS to request authorization. **3.** Upon receipt of AuthorizeRequest, the CSMS responds with AuthorizeResponse. This response message includes the GroupId. **4.** The Charging Station stores the GroupIdToken with the authorization information of EV Driver 1. **5.** EV Driver 2 presents an IdToken. **6.** The Charging Station sends AuthorizeRequest to the CSMS to request authorization. **7.** Upon receipt of AuthorizeRequest, the CSMS responds with AuthorizeResponse. This response message includes the GroupId. **8.** Based on the matching GroupId information in both responses, the Charging Station authorizes the action.
5	**Prerequisite(s)**	EV Driver 1 and EV Driver 2 have the same GroupId.
6	**Postcondition(s)**	GroupId is known by the Charging Station.
7	**Error handling**	n/a
8	**Remark(s)**	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. The groupId mechanism as described in this use case also works when using the Authorization Cache, as the groupId is stored in the cache.

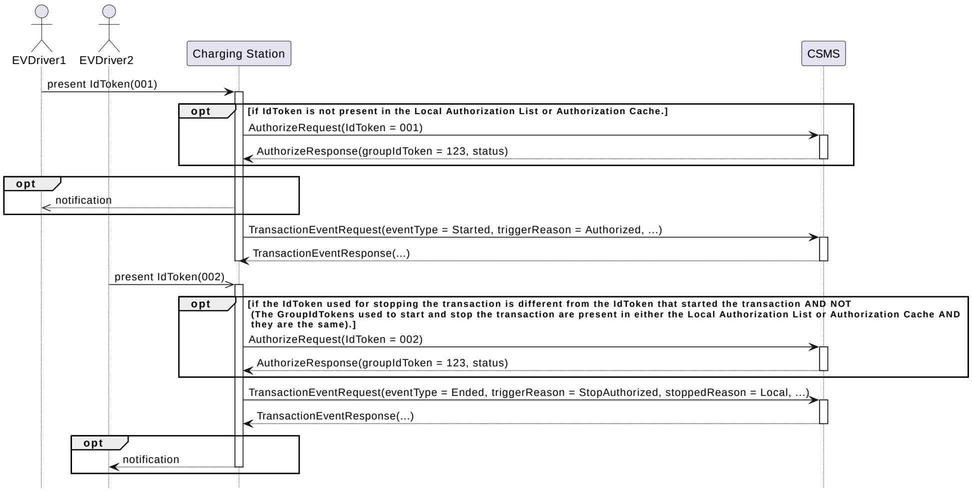

EVDriver1
EVDriver2
Charging Station
CSMS
present IdToken(001)
opt
[if IdToken is not present in the Local Authorization List or Authorization Cache.]
AuthorizeRequest(IdToken = 001)
AuthorizeResponse(groupIdToken = 123, status)
opt
notification
TransactionEventRequest(eventType = Started, triggerReason = Authorized, ...)
TransactionEventResponse(...)
present IdToken(002)
opt
[if the IdToken used for stopping the transaction is different from the IdToken that started the transaction AND NOT
(The GroupIdTokens used to start and stop the transaction are present in either the Local Authorization List or Authorization Cache AND
they are the same).]
AuthorizeRequest(IdToken = 002)
AuthorizeResponse(groupIdToken = 123, status)
TransactionEventRequest(eventType = Ended, triggerReason = StopAuthorized, stoppedReason = Local, ...)
TransactionEventResponse(...)
opt
notification

Figure 30. Sequence Diagram: Authorization by GroupId

C09 - Authorization by GroupId - Requirements

Table 51. 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 *groupIdToken* element in IdTokenInfo

ID	Precondition	Requirement definition
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 groupIdToken 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 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
C09.FR.09	If the IdToken in AuthorizeRequest has an associated groupIdToken	AuthorizeResponse from CSMS SHALL include groupIdToken.
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 groupIdToken	TransactionEventResponse from CSMS SHALL include groupIdToken.
C09.FR.13		The field *idToken.type* of a GroupIdToken SHOULD be `Central`

2.4. Authorization Cache

C10 - Store Authorization Data in the Authorization Cache

No.	Type	Description
1	**Name**	Store Authorization Data in the Authorization Cache
2	**ID**	C10
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	**1.** The Charging Station receives a AuthorizeResponse, ReserveNowRequest or TransactionEventResponse response message from the CSMS. **2.** The Cache is updated by the Charging Station using all received IdTokenInfo from the response message from the CSMS.
	Alternative scenario(s)	n/a
5	**Prerequisite(s)**	An Authorization Cache is implemented and and the value of the `AuthCacheEnabled` Configuration Variable is set to 'true'.
6	**Postcondition(s)**	**Successful postcondition:** The Charging Station stored the newly received IdTokenInfo data in the Authorization Cache. **Failure postcondition:** The Charging Station was *not* able to store the Authorization Cache.
7	**Error handling**	n/a
8	**Remark(s)**	n/a

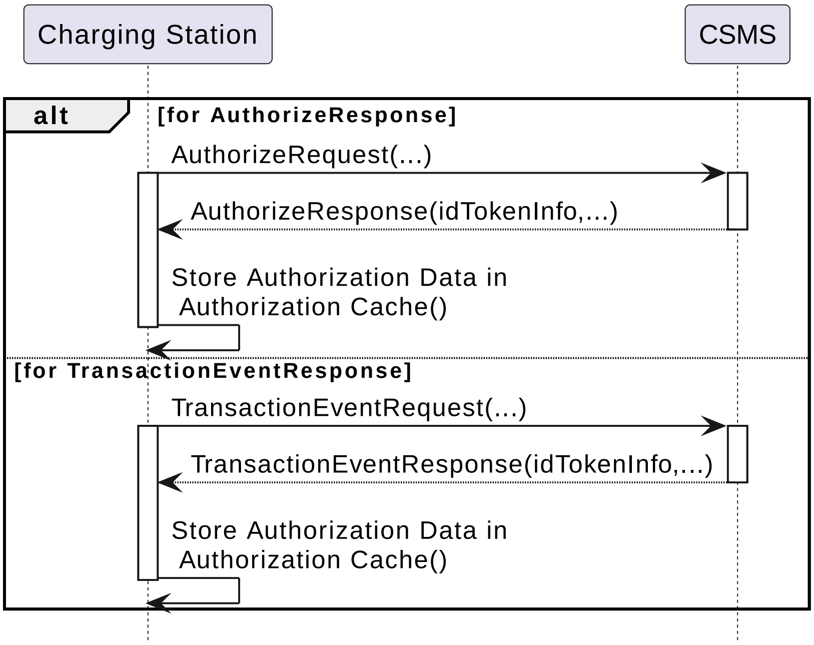

Charging Station
CSMS
alt
[for AuthorizeResponse]
AuthorizeRequest(...)
AuthorizeResponse(idTokenInfo,...)
Store Authorization Data in Authorization Cache()
[for TransactionEventResponse]
TransactionEventRequest(...)
TransactionEventResponse(idTokenInfo,...)
Store Authorization Data in Authorization Cache()

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

C10 - Store Authorization Data in the Authorization Cache - Requirements

Table 52. C10 - Requirements

ID	Precondition	Requirement definition	Note
C10.FR.01		The Authorization Cache SHALL contain all the latest received identifiers (regardless of their status).	

ID	Precondition	Requirement definition	Note
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.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	When IdTokenInfoType does not contain a value for *cacheExpiryDateTime*	The time a token is considered to be present 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.10	NOT C10.FR.13 AND when more than `AuthCacheLifeTime` seconds have passed since *idTokenInfo* was last updated	The Authorization Cache entry SHALL be removed from the cache or changed to `Expired`.	A *cacheExpiryDateTime* in the past will prevent an idToken from being stored in the authorization cache, or remove it from authorization cache if it was already present. This is used e.g. for prepaid accounts that should not be kept in authorization cache.
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.
C10.FR.13	When IdTokenInfoType contains a value for *cacheExpiryDateTime* and current time is greater than *idTokenInfo.cacheExpiryDateTime*	The Authorization Cache entry SHALL be removed from the cache or changed to `Expired`.	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.14 **(2.1)**	C10.FR.04 OR C10.FR.05	Charging Station SHALL store all IdTokenType fields except for the *additionalInfo* field, together with all fields of IdTokenInfoType from AuthorizeResponse or TransactionEventResponse, except for the *additionalInfo* field in *groupIdToken*.	*additionalInfo* of *idToken* or *groupIdToken* is not cached.

C11 - Clear Authorization Data in Authorization Cache

No.	Type	Description
1	**Name**	Clear Authorization Data in Authorization Cache
2	**ID**	C11
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	**1.** The CSMS requests the Charging Station to clear its Authorization Cache by sending ClearCacheRequest. **2.** The Charging Station responds with the status *Accepted*.
5	**Prerequisite(s)**	Authorization Cache is supported and enabled by the `AuthCacheEnabled` Configuration Variable.
6	**Postcondition(s)**	**Successful postcondition:** The Charging Station *Successfully* cleared the Authorization Cache. **Failure postcondition:** The Charging Station was *not* able to clear the Authorization Cache.
7	**Error handling**	n/a
8	**Remark(s)**	n/a

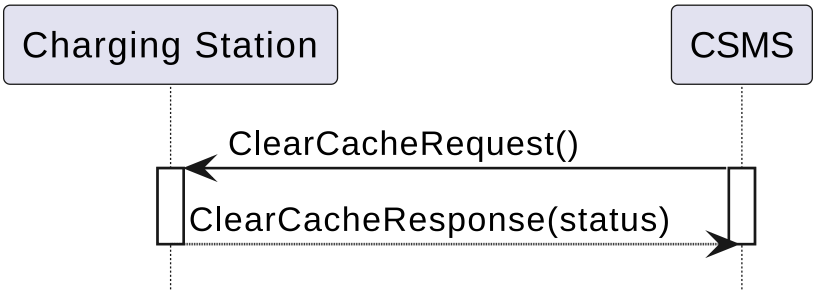


Figure 32. Sequence Diagram: Clear Authorization Data in Authorization Cache

C11 - Clear Authorization Data in Authorization Cache - Requirements

Table 53. 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

No.	Type	Description
1	**Name**	Start Transaction - Cached Id
2	**ID**	C12
3	**Objective(s)**	To enable the EV Driver to *Online* 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	**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)**	`AuthCacheEnabled` = true `LocalPreAuthorize` = true The Id of the EV Driver is Cached in the Authorization Cache Id is valid
6	**Postcondition(s)**	**Successful postcondition:** The EV Driver is authorized to start a transaction by using the Authorization Cache. **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.
7	**Error handling**	When the Charging Station has an IdToken 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 IdTokenInfo in the TransactionEventResponse which contains the newer invalid status. What happens in such a cases depends on the Configuration Variables: `MaxEnergyOnInvalidId` and `StopTxOnInvalidId`.
8	**Remark(s)**	If the Charging Station has implemented an Authorization Cache, then upon receipt of a AuthorizeResponse message the Charging Station updates the Cache entry. For a Cached valid IdToken it is not logical to send AuthorizeRequest. The TransactioneventResponse message also contains the IdToken information. If the IdToken has become no longer valid, the Charging Station will learn this from this TransactioneventResponse. So if the IdToken is no longer valid, the Charging Station might decide to stop the energy offering, and depending on the configuration even stop the transaction. The scenario description and sequence diagram above are based on the Configuration Variable for start transaction being configured as follows: `TxStartPoint`: 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 send. For more details see the use case: E01 - Start Transaction options.

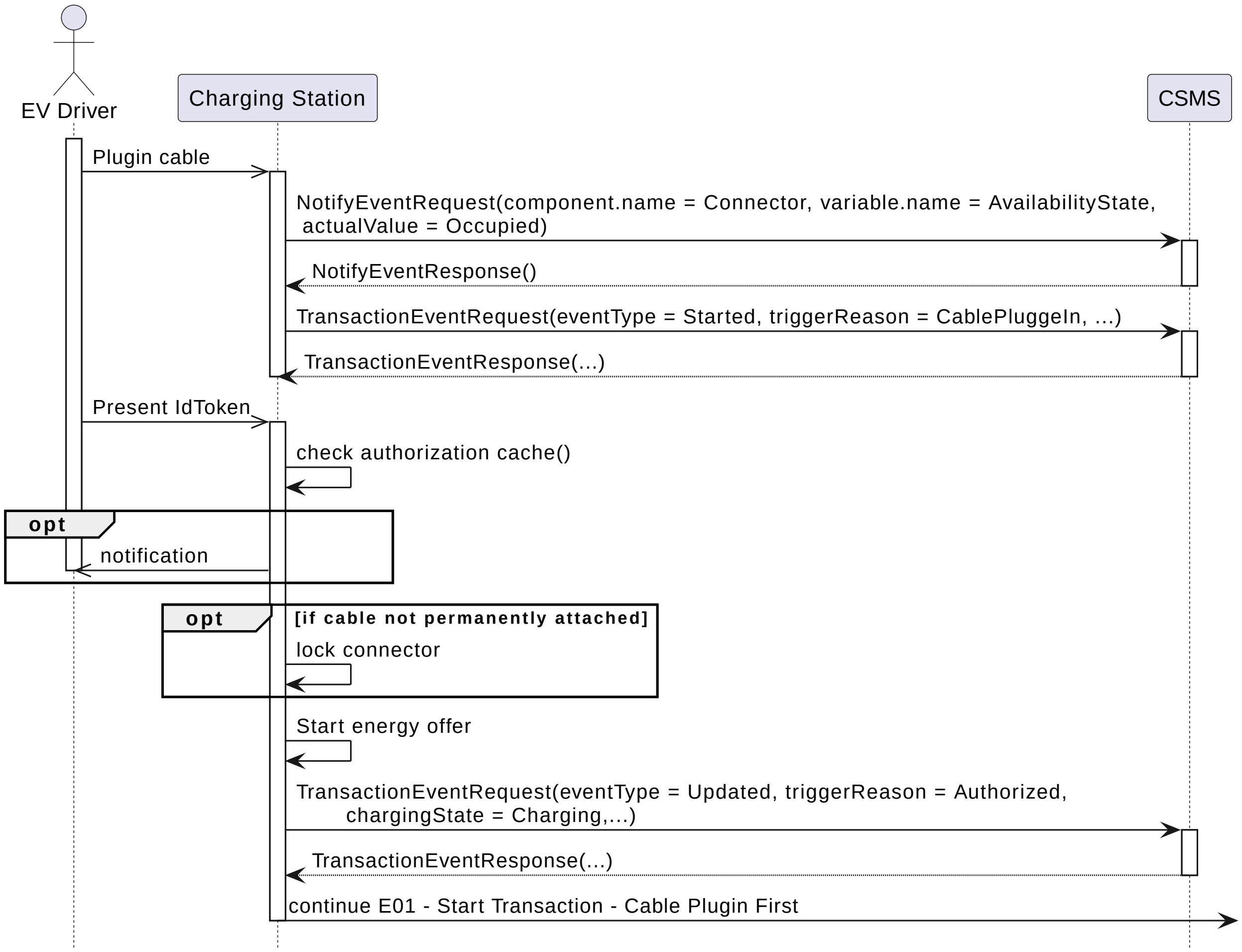

EV Driver
Charging Station
CSMS
Plugin cable
NotifyEventRequest(component.name = Connector, variable.name = AvailabilityState, actualValue = Occupied)
NotifyEventResponse()
TransactionEventRequest(eventType = Started, triggerReason = CablePluggeIn, ...)
TransactionEventResponse(...)
Present IdToken
check authorization cache()
opt
notification
opt
[if cable not permanently attached]
lock connector
Start energy offer
TransactionEventRequest(eventType = Updated, triggerReason = Authorized, chargingState = Charging,...)
TransactionEventResponse(...)
continue E01 - Start Transaction - Cable Plugin First

Figure 33. Sequence Diagram: Start Transaction - Cached Id

C12 - Start Transaction - Cached Id - Requirements

Table 54. C12 - Requirements

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

No.	Type	Description
1	**Name**	Offline Authorization through Local Authorization List

No.	Type	Description
2	**ID**	C13
3	**Objective(s)**	To authorize an idToken by using the Local Authorization List while *Offline*.
4	**Description**	This use case describes how to authorize an IdToken, when communication with the CSMS is not possible. 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.
	Actors	EV Driver, Charging Station
	Scenario description	**1.** The Charging Station is *Offline* **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)**	*Local Authorization List* is available *Local Authorization List* is enabled via `LocalAuthListEnabled` Charging Station is *Offline* The Id of the EV Driver is in the *Local Authorization List* Id is valid
6	**Postcondition(s)**	**Successful postcondition:** The Charging Station accepts tokens on the Local Authorization List when it is offline. **Failure postcondition:** The Charging Station does not accept tokens on the Local Authorization List when it is offline.
7	**Error handling**	n/a
8	**Remark(s)**	n/a

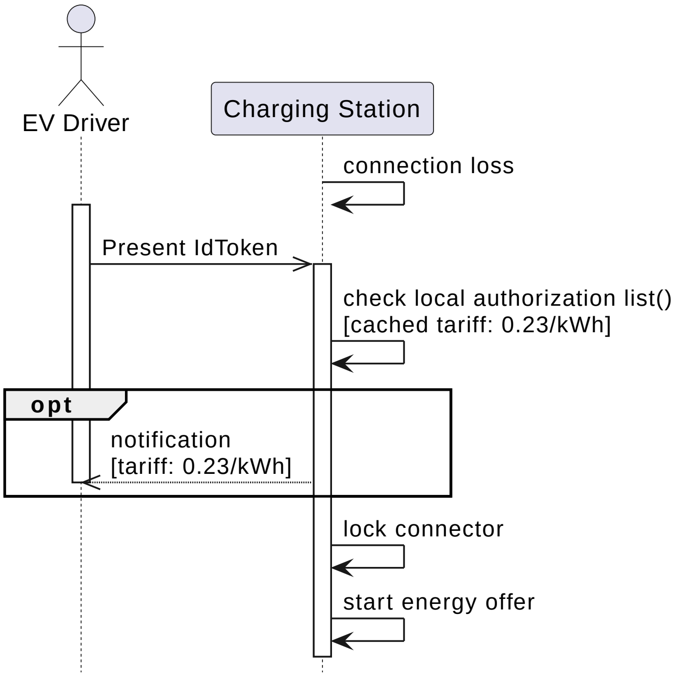


Figure 34. Sequence Diagram: Offline Authorization through Local Authorization List

C13 - Offline Authorization through Local Authorization List - Requirements

Table 55. 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.	

ID	Precondition	Requirement definition	Note
C13.FR.02	If configuration variable `OfflineTxForUnknownIdEnabled` is false AND The Charging Station is offline AND `LocalAuthListSupportsExpiryDateTime` does not exist or is false	Only identifiers that are present in a Local Authorization List that have a status *Accepted* SHALL be allowed to authorize a transaction.	This means that Charging Station does not check for *cacheExpiryDateTime*.
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 AND `LocalAuthListSupportsExpiryDateTime` does not exist or is false	Any identifier that is present in neither the Authorization Cache nor the Local Authorization List SHALL be allowed to authorize a transaction AND any identifiers that are present in a Local Authorization List that have a status *Accepted* SHALL be allowed to authorize a transaction.	This means that Charging Station does not check for *cacheExpiryDateTime*. See also C15.FR.08
C13.FR.05	If configuration variable `OfflineTxForUnknownIdEnabled` is false AND The Charging Station is offline AND `LocalAuthListSupportsExpiryDateTime` = true	Only identifiers that are present in a Local Authorization List that have a status *Accepted* and for which *cacheExpiryDateTime* has not passed SHALL be allowed to authorize a transaction.	When *cacheExpiryDateTime* is absent, the idToken will not expire in Local Authorization List.
C13.FR.06	If configuration variable `OfflineTxForUnknownIdEnabled` is true AND The Charging Station is offline AND `LocalAuthListSupportsExpiryDateTime` = true	Any identifier that is present in neither the Authorization Cache nor the Local Authorization List SHALL be allowed to authorize a transaction AND any identifiers that are present in a Local Authorization List that have a status *Accepted* and for which *cacheExpiryDateTime* has not passed SHALL be allowed to authorize a transaction.	This means that an expired token in the Local Authorization List is not authorized, because it is not an "unknown id".

C14 - Online Authorization through Local Authorization List

No.	Type	Description
1	**Name**	Online Authorization through Local Authorization List
2	**ID**	C14
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
	Scenario description	**1.** The EV Driver presents IdToken **2.** The Charging Station checks if the IdToken is known and has status *Accepted* in the Local Authorization List. **3.** If the IdToken is not known, or the IdToken is not *Accepted* the Charging Station sends an AuthorizeRequest **4.** The Charging Station starts charging.
5	**Prerequisite(s)**	*Local Authorization List* is available *Local Authorization List* is enabled via `LocalAuthListEnabled` The Id of the EV Driver is in the *Local Authorization List* Id is valid `LocalPreAuthorize` is set to *true*
6	**Postcondition(s)**	**Successful postcondition:** The Charging Station accepts tokens on the Local Authorization List. **Failure postcondition:** The Charging Station does not accept tokens on the Local Authorization List.
7	**Error handling**	n/a
8	**Remark(s)**	n/a

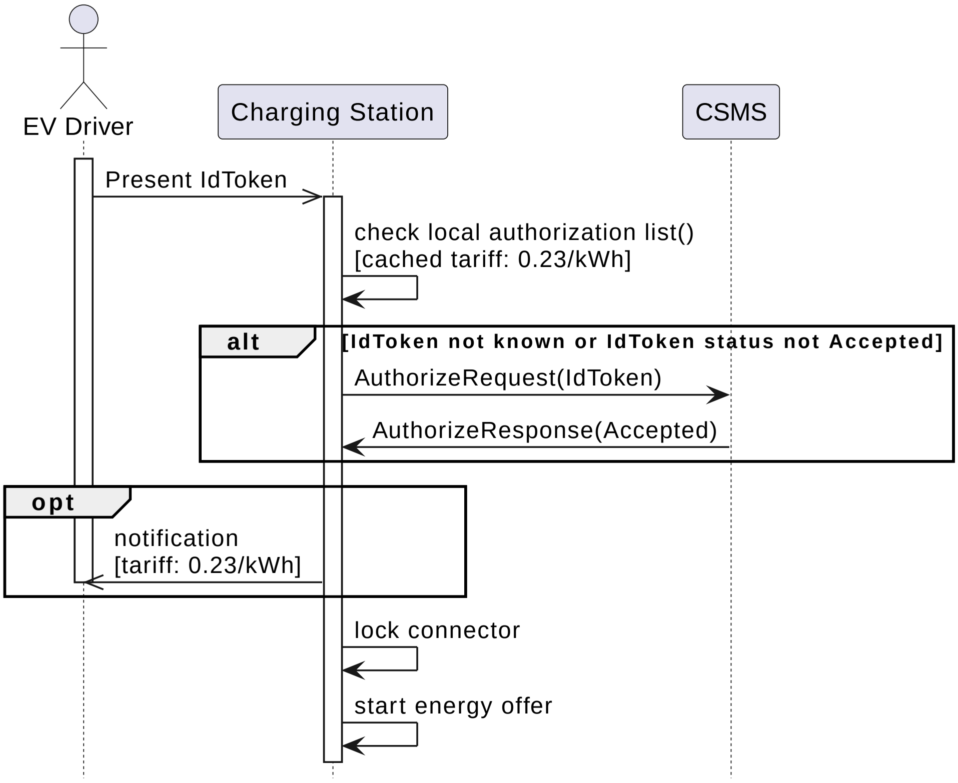

EV Driver
Charging Station
CSMS
Present IdToken
check local authorization list()
[cached tariff: 0.23/kWh]
alt
[IdToken not known or IdToken status not Accepted]
AuthorizeRequest(IdToken)
AuthorizeResponse(Accepted)
opt
notification
[tariff: 0.23/kWh]
lock connector
start energy offer

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

C14 - Online Authorization through Local Authorization List - Requirements

Table 56. C14 - Requirements

ID	Precondition	Requirement definition	Note
C14.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.	Same as C13.FR.01
C14.FR.02	Identifier presented is in the Local Authorization List with a *status* `Accepted` AND `LocalAuthListSupportsExpiryDateTime` does not exist or is false	The Charging Station SHALL start charging without sending an AuthorizeRequest.	This means that Charging Station does not check for *cacheExpiryDateTime*.
C14.FR.03	Identifiers presented is in the Local Authorization List with a status OTHER than *Accepted*	The Charging Station SHALL send an AuthorizeRequest to try to authorize this IdToken.	
C14.FR.04	Identifier presented is in the Local Authorization List with a *status* `Accepted` AND `LocalAuthListSupportsExpiryDateTime` = true AND the *cacheExpiryDateTime* has not passed	The Charging Station SHALL start charging without sending an AuthorizeRequest.	When *cacheExpiryDateTime* is absent, the idToken will not expire in Local Authorization List.
C14.FR.05	Identifier presented is in the Local Authorization List with a *status* `Accepted` AND `LocalAuthListSupportsExpiryDateTime` = true AND the *cacheExpiryDateTime* has passed	The Charging Station SHALL send an AuthorizeRequest to try to authorize this IdToken.	IdToken will be disregarded, as if not present in Local Authorization List, when *cacheExpiryDateTime* has passed.

2.6. Offline Authorization

C15 - Offline Authorization of unknown Id

No.	Type	Description
1	**Name**	Offline Authorization of unknown Id
2	**ID**	C15
	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	**1.** The EV Driver wants to start charging the EV and presents the IdToken. **2.** The Charging Station checks the Authorization Cache, the IdToken is not present in the Authorization Cache. **3.** The Charging Station checks the Local Authorization List, the IdToken is not present in the Local Authorization List. **4.** The Charging Station accepts the unknown IdToken if `OfflineTxForUnknownIdEnabled` is set *True* **5.** The Charging Station rejects the unknown IdToken if `OfflineTxForUnknownIdEnabled` is set *False*
	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 *Offline*. Unknown IdToken presented (Not in the Authorization Cache and/or Local Authorization List).
6	**Postcondition(s)**	**Successful postcondition:** The authorization status in TransactionEventResponse, that is received once the Charging Station is online, is *Accepted*. **Failure postcondition:** The authorization status in TransactionEventResponse that is received once the Charging Station is online, is *not Accepted*, although `OfflineTxForUnknownIdEnabled` is *True*.
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.

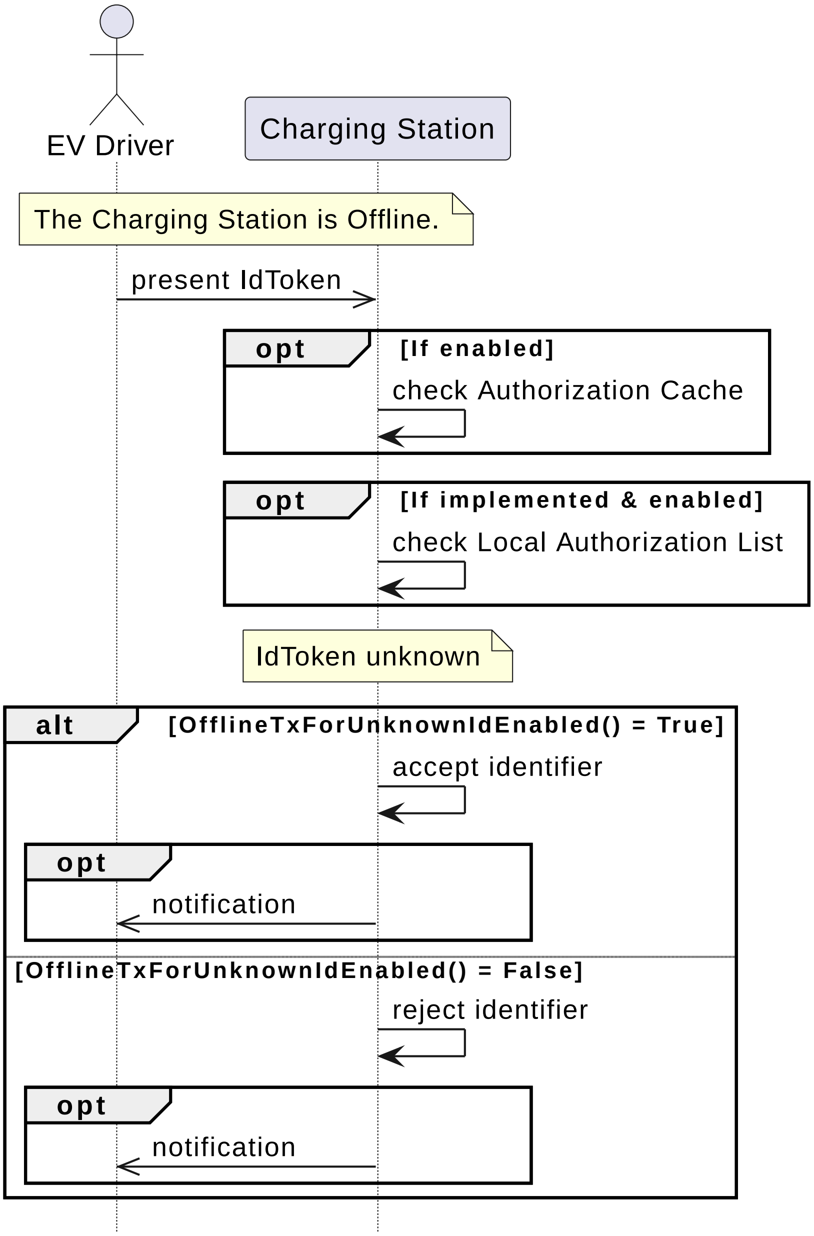

EV Driver
Charging Station
The Charging Station is Offline.
present IdToken
opt
[If enabled]
check Authorization Cache
opt
[If implemented & enabled]
check Local Authorization List
IdToken unknown
alt
[OfflineTxForUnknownIdEnabled() = True]
accept identifier
opt
notification
[OfflineTxForUnknownIdEnabled() = False]
reject identifier
opt
notification

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

C15 - Offline Authorization of unknown Id - Requirements

Table 57. C15 - Requirements

ID	Precondition	Requirement definition	Note
C15.FR.01	If the identifier is authorized via `OfflineTxForUnknownIdEnabled`	The Charging Station SHALL NOT add the token to Authorization Cache	
C15.FR.02	When connection to the CSMS is restored	The Charging Station SHALL send a TransactionEventRequest for any transaction that was authorized *offline*.	As explained in transaction-related message handling
C15.FR.03 **(2.1)**	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 `EVConnected`.	The use of *chargingState* `SuspendedEVSE` instead of `EVConnected` in this situation has been deprecated as of OCPP 2.1.

ID	Precondition	Requirement definition	Note
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

No.	Type	Description
1	**Name**	Stop Transaction with a Master Pass
2	**ID**	C16
3	**Objective(s)**	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

No.	Type	Description
	Scenario description	**1.** The User (Law Enforcement official) presents his IdToken at the Charging Station. **2.** The Charging Station sends AuthorizeRequest to the CSMS to request authorization. **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. **4a.** If the Charging Station has a UI, then the Charging Station "Shows" the Master Pass UI. **5a.** The user selects which transactions to stop. **6a.** The Charging Station stops the selected transaction(s) AND sends a TransactionEventRequest (eventType = Ended, stopReason = MasterPass) to the CSMS for every stopped transaction. **7a.** Upon receipt of TransactionEventRequest the CSMS responds with TransactionEventResponse. **4b.** If the Charging Station does NOT have a UI, then the Charging Station stops all transactions AND sends a TransactionEventRequest (eventType = Ended, stopReason = MasterPass) to the CSMS for every stopped transaction. **5b.** Upon receipt of TransactionEventRequest the CSMS responds with TransactionEventResponse.
	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.
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**	The scenario description and sequence diagram above are based on the Configuration Variable for stop transaction being configured as follows. `TxStopPoint`: Authorized, DataSigned, PowerPathClosed, EnergyTransfer 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

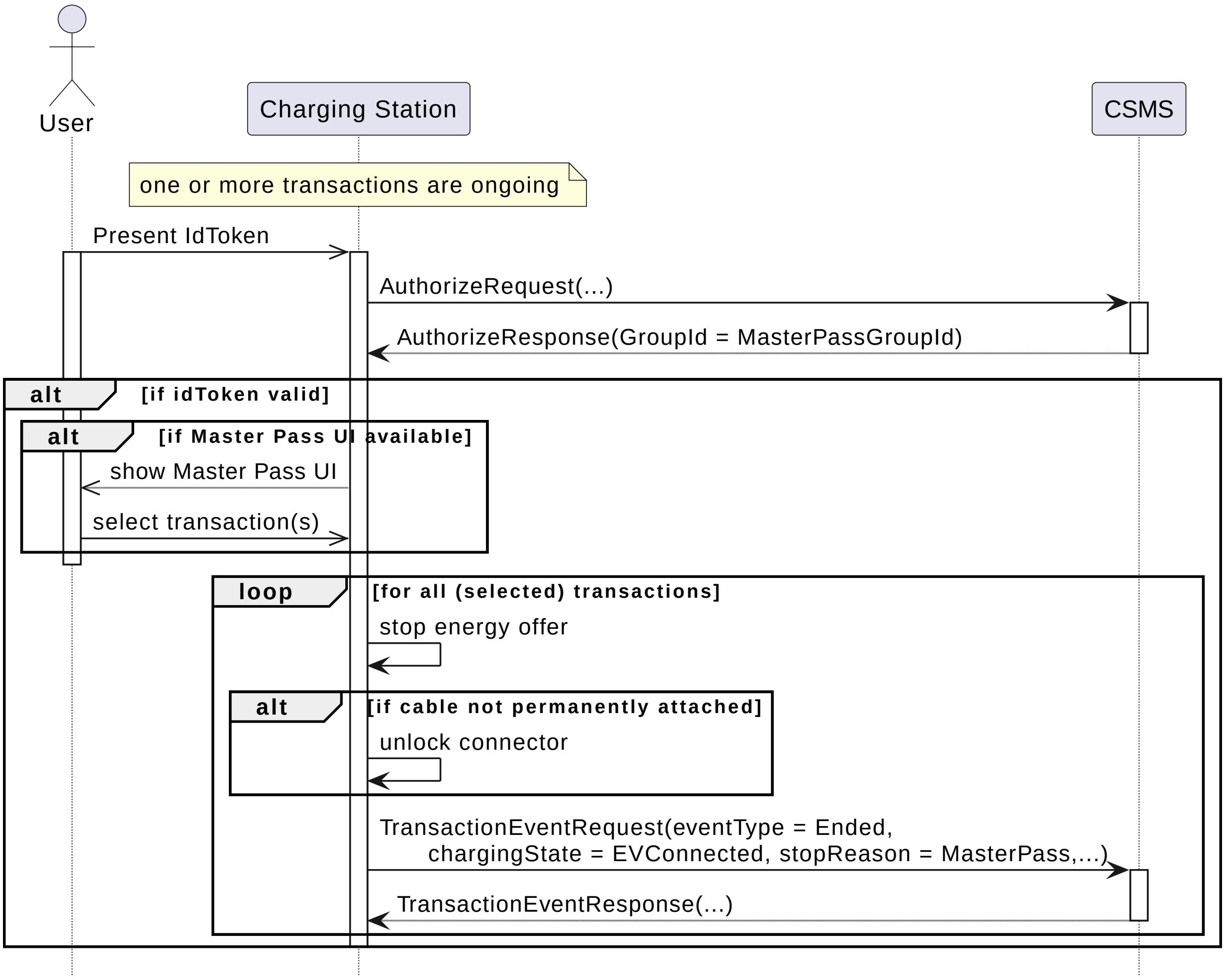

User
Charging Station
CSMS
one or more transactions are ongoing
Present IdToken
AuthorizeRequest(...)
AuthorizeResponse(GroupId = MasterPassGroupId)
alt [if idToken valid]
alt [if Master Pass UI available]
show Master Pass UI
select transaction(s)
loop [for all (selected) transactions]
stop energy offer
alt [if cable not permanently attached]
unlock connector
TransactionEventRequest(eventType = Ended,
chargingState = EVConnected, stopReason = MasterPass,...)
TransactionEventResponse(...)

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

C16 - Stop Transaction with a Master Pass - Requirements

Table 58. C16 - 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 with input capabilities.	The Charging Station SHALL "show" the Master Pass UI to let user select which transaction to stop.
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 as described in use case E07.
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.
C16.FR.07	C16.FR.01	Charging Station SHALL stop the transaction as described in use case E07.
C16.FR.08	C16.FR.02 OR C16.FR.07	Charging Station SHALL set *transactionInfo.stoppedReason* = `MasterPass` in TransactionEventRequest with *eventType* = `Ended`.

2.8. Authorization with prepaid card

C17 - Authorization with prepaid card

New in OCPP 2.1

No.	Type	Description
1	**Name**	Authorization with prepaid card
2	**ID**	C17
3	**Objective(s)**	To enable the Charging Station to request the CSMS to authorize an EV Driver to start charging based on a prepaid balance.
4	**Description**	An EV Driver uses a prepaid card for charging. Charging Station ensures that no more energy is charged, than is covered by the prepaid balance.
	Actors	Charging Station, CSMS, EV Driver
	Scenario description	*Authorization causes transaction to start, e.g. TxStartPoint = "Authorized", or "PowerPathClosed" and cable is already connected* 1. EV Driver presents an RFID card that is linked to an account with a prepaid balance. 2. Charging Station sends AuthorizeRequest to CSMS with *idToken* that is read from the card. 3. If card is not valid, CSMS responds with AuthorizeResponse with *idTokenInfo.status* = `Invalid`. a. Charging station does not start a transaction. b. Use case ends. 4. If card is valid, but prepaid balance is not positive, CSMS responds with AuthorizeResponse with *idTokenInfo.status* = `NoCredit` and *idTokenInfo.cacheExpiryDateTime* set to now. a. Charging station does not start a transaction. b. Use case ends. 5. If card is valid and prepaid balance is positive, CSMS responds with AuthorizeResponse with *idTokenInfo.status* = `Accepted` and *idTokenInfo.cacheExpiryDateTime* set to now. a. Charging station sends a TransactionEventRequest with *eventType* = `Started`, with *idToken* of the prepaid card b. CSMS responds with a TransactionEventResponse with *transactionLimit.maxCost* set to amount of credit left in account. c. Behavior according to use case E16 - Transactions with fixed cost, energy or time.

No.	Type	Description
	Scenario description #2	*Transaction already started when authorizing, e.g. TxStartPoint = "EVConnected" and cable is connected* 1. EV Driver presents an RFID card that is linked to an account with a prepaid balance. 2. Charging Station sends AuthorizeRequest to CSMS with *idToken* that is read from the card. 3. If card is not valid, CSMS responds with AuthorizeResponse with *idTokenInfo.status* = `Invalid` a. Charging station does not start energy transfer. b. Use case ends. (Transaction will be stopped when EV Driver disconnects the cable). 4. CSMS checks prepaid balance of associated account. 5. If card is valid, but prepaid balance is not positive, CSMS responds with AuthorizeResponse with *idTokenInfo.status* = `NoCredit` and *idTokenInfo.cacheExpiryDateTime* set to now. a. Charging station does not start energy transfer. b. Use case ends. (Transaction will be stopped when EV Driver disconnects the cable). 6. If card is valid and prepaid balance is positive, CSMS responds with AuthorizeResponse with *idTokenInfo.status* = `Accepted` and *idTokenInfo.cacheExpiryDateTime* set to now. a. Charging station sends a TransactionEventRequest with *eventType* = `Updated`, with *idToken* of the prepaid card. b. CSMS responds with a TransactionEventResponse with *transactionLimit.maxCost* set to amount of credit left in account. c. Behavior according to use case E16 - Transactions with fixed cost, energy or time.
	Scenario description #3	*Transaction not started by authorizing, e.g. TxStartPoint = "EVConnected" or "PowerPathClosed", and cable is not yet connected* 1. EV Driver presents an RFID card that is linked to an account with a prepaid balance. 2. Charging Station sends AuthorizeRequest to CSMS with *idToken* that is read from the card. 3. If card is not valid, CSMS responds with AuthorizeResponse with *idTokenInfo.status* = `Invalid` a. Use case ends. 4. CSMS checks prepaid balance of associated account. 5. If card is valid, but prepaid balance is not positive, CSMS responds with AuthorizeResponse with *idTokenInfo.status* = `NoCredit` and *idTokenInfo.cacheExpiryDateTime* set to now. a. Use case ends. 6. If card is valid and prepaid balance is positive, CSMS responds with AuthorizeResponse with *idTokenInfo.status* = `Accepted` and *idTokenInfo.cacheExpiryDateTime* set to now. 7. If EV Driver connects cable within timeout (EVConnectionTimeout), then a. Charging station sends a TransactionEventRequest with *eventType* = `Started`, with *idToken* of the prepaid card. b. CSMS responds with a TransactionEventResponse with *transactionLimit.maxCost* set to amount of credit left in account. c. Behavior according to use case E16 - Transactions with fixed cost, energy or time.
5	**Prerequisite(s)**	CSMS supports prepaid cards.
6	**Postcondition(s)**	Energy transfer is stopped when amount of prepaid balance is reached.
7	**Error handling**	

No.	Type	Description
8	**Remark(s)**	Prepaid *idTokens* shall not be cached, because an AuthorizeRequest is needed to get the current balance. *CacheExpiryDateTime* is set to current time so that *idToken* does not remain in authorization cache. This is a version of use case C01 - EV Driver Authorization using RFID in which CSMS may return an authorization status `NoCredit`. The transaction with a *maxCost* limit is described in E16 - Transactions with fixed cost, energy, SoC or time.

NOTE Following diagram only shows parameters related to the use case. More parameters are required in these messages.

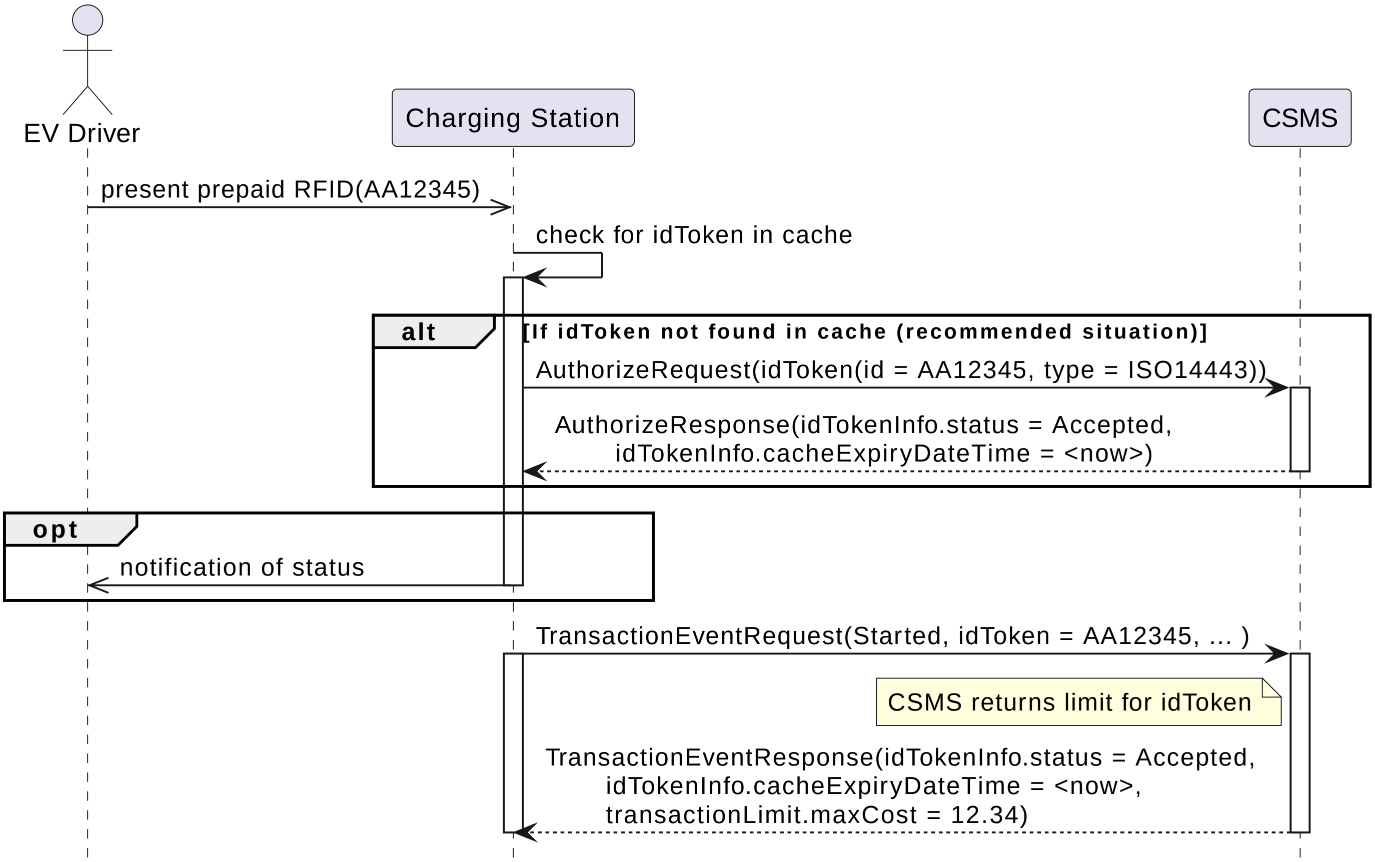

EV Driver
Charging Station
CSMS
present prepaid RFID(AA12345)
check for idToken in cache
alt
[If idToken not found in cache (recommended situation)]
AuthorizeRequest(idToken(id = AA12345, type = ISO14443))
AuthorizeResponse(idTokenInfo.status = Accepted,
idTokenInfo.cacheExpiryDateTime = <now>)
opt
notification of status
TransactionEventRequest(Started, idToken = AA12345, ...)
CSMS returns limit for idToken
TransactionEventResponse(idTokenInfo.status = Accepted,
idTokenInfo.cacheExpiryDateTime = <now>,
transactionLimit.maxCost = 12.34)

Figure 38. Sequence diagram prepaid authorization

C17 - EV Driver Authorization using prepaid card - Requirements

Table 59. C17 - Requirements

ID	Precondition	Requirement definition	Note
C17.FR.01	CSMS receives an AuthorizeRequest for an *idToken* that is valid and associated with a prepaid account that has a positive balance	CSMS SHALL respond with an AuthorizeResponse with *idTokenInfo* with *status* = `Accepted` and *cacheExpiryDateTime* set to current date/time.	Prepaid token is not allowed to remain in the local authorization cache.
C17.FR.02	CSMS receives an AuthorizeRequest for an *idToken* that is valid and associated with a prepaid account that does not have a positive balance	CSMS SHALL respond with an AuthorizeResponse with *idTokenInfo* that has *status* = `NoCredit` and *cacheExpiryDateTime* set to current date/time.	Prepaid token is not allowed to remain in the local authorization cache.
C17.FR.03	CSMS receives an TransactionEventRequest with an *idToken* that is valid and associated with a prepaid account that has a positive balance	CSMS SHALL include (only one time) the field *transactionLimit.maxCost*, set to the maximum amount that may be spent, in the TransactionEventResponse	See also E16 - Transactions with fixed cost, energy, SoC or time.
C17.FR.04	C17.FR.01	Charging Station SHALL offer energy	Authorization status of *idToken* is `Accepted`. See also C01 - EV Driver Authorization using RFID.
C17.FR.05	C17.FR.02	Charging Station SHALL NOT offer energy	Authorization status of *idToken* is NOT `Accepted`. See also C01 - EV Driver Authorization using RFID.

NOTE Other requirements from C. Authorization also apply, notably C01 - EV Driver Authorization using RFID

2.9. Ad hoc payments

Ad hoc payment or direct payment refers to payment of a charging sesssion with a debit or credit card. This requires a built-in or stand-alone payment terminal or site from a payment service provider (PSP) that the user can access with a mobile device. The payment terminal is a secure, closed box, that has its own secure connection to a payment service provider (PSP).

Integrated payment terminals

Most use cases in this chapter are about an integrated payment terminal that is part of the Charging Station and provides authorization for a valid payment card to start a charging session. The payment terminal provides an API to communicate internally with the Charging Station. These are proprietary APIs that are not described in this document. The payment terminal requests authorization of the payment card via its own direct connection to the PSP.

The use cases C18 - Authorization using locally connected payment terminal to C23 - Increasing authorization amount describe ad hoc payment via an integrated payment terminal.

Stand-alone payment terminals

The payment terminal can also be a separate unit that is serving several Charging Stations. The latter is called a stand-alone payment terminal or payment kiosk in this specification, because it has no direct connection with the Charging Station(s). Once a payment card has been authorized, the stand-alone payment terminal will instruct CSMS to remotely start a charging transaction on a specified Charging Station.

This scenario is described in use case C24 - Ad hoc payment via stand-alone payment terminal. The interface between a stand-alone payment terminal and CSMS is not described in this specification, because it is not related to Charging Station communication.

Ad hoc via QR code

Charging Station that do not have an integrated or stand-alone payment terminal can still support ad hoc payment via a payment website. A common approach for this is to show a QR code on the Charging Station that provides the URL of the payment website, which the user can access using a mobile device. Once payment details are entered and validated, CSMS remotely starts a charging session for the user. In order to prevent fraud by pasting fake QR stickers on the Charging Station it is recommended to show a dynamic QR code on the Charging Station display.

This is described in use case C25 - Ad hoc payment via a QR code.

C18 - Authorization using locally connected payment terminal

New in OCPP 2.1

No.	Type	Description
1	**Name**	Authorization using locally connected payment terminal
2	**ID**	C18
3	**Objective**	To start/authorize a transaction from a payment terminal connected directly to the Charging Station.
4	**Description**	EV Driver presents a payment card to integrated payment terminal in Charging Station. Payment terminal authorizes payment card with Payment Service Provider. The PspRef from PSP is used as *idToken* with *type* = `DirectPayment`, card details are added in *idToken.additionalInfo* fields.
	Actors	EV Driver, Payment Terminal, PSP, CSMS, Charging Station
	Scenario description	*PaymentCtrlr.AuthorizeDirectPayment = false* *(A default ad hoc tariff exists on Charging Station)* 1. EV Driver presents payment card to payment terminal. 2. Payment terminal requests PSP for authorization of the amount configured in PaymentCtrlr.Authorization amount. 3. Upon approval PSP returns an approved status and a unique reference for this payment in the PSP platform. a. This unique reference is called "PspRef" in OCPP. 4. Payment Terminal sends payment details, such as PspRef, cardBin, cardLast4Digits, etc. to Charging Station controller. a. Prior to this use case the fields that need to be added as *additionalInfo* elements have been configured by CSMS in the actual *value* of PaymentCtrlr.PaymentDetails. b. The fields that Payment Terminal is able to provide have been reported in a device model report by Charging Station in the *valuesList* of PaymentCtrlr.PaymentDetails. 5. Charging Station authorizes *idToken* with value <PspRef> and *type* = `DirectPayment`. 6. When EV Driver plugs-in vehicle, Charging Station starts a transaction and sends a TransactionEventRequest with *eventType* = `Started` and *idToken* with the payment details in *additionalInfo* element of IdTokenType, and with *transactionLimit.maxCost* set to the payment card authorization amount, as configured in PaymentCtrlr.AuthorizationAmount. a. CSMS responds with TransactionEventResponse.
	Alternative scenario(s)	*PaymentCtrlr.AuthorizeDirectPayment = true* **Steps 1 to 4 identical to scenario above** 5. Charging Station sends AuthorizeRequest with *idToken* with value <PspRef> and *type* = `DirectPayment` and payment details in *additionalInfo* element of IdTokenType. a. CSMS responds with AuthorizeResponse with *status* = `Accepted` and optionally a tariff to use (in case a default tariff is not applicable). 6. When EV Driver plugs in vehicle, Charging Station starts a transaction and sends a TransactionEventRequest with *eventType* = `Started` and *idToken* with the payment details in *additionalInfo* element of IdTokenType, and with *transactionLimit.maxCost* set to the payment card authorization amount, as configured in PaymentCtrlr.AuthorizationAmount. a. CSMS responds with TransactionEventResponse.

No.	Type	Description
5	**Prerequisites**	The Charging Station has a payment terminal which is directly connected to the Charging Station. An ad hoc tariff has been set as the default tariff as described in use case I07 - Local Cost Calculation - Set Default Tariff.
6	**Postcondition(s)**	EV Driver is authorized to charge.
7	**Error handling**	When the payment service provider does not authorize the payment card, or when the payment terminal fails to communicate, then EV Driver is not authorized by Charging Station. This means that when PaymentCtrlr.AuthorizeDirectPayment = true Charging Station will not send an AuthorizeRequest.
8	**Remark(s)**	The communication between PSP and payment terminal, and the API between payment terminal and Charging Station are only described at an abstract level in this use case. PspRef is the reference from PSP that is used within PSP platform and is used to settle the payment. A payment terminal may also have its own transaction or session ID, but for the OCPP use case only the reference that is needed to settle the payment is relevant. This use case assumes a default tariff and support for local cost calculation. It can also work when cost calculation is performed by CSMS, but in that case it will not be possible to stop exactly a the *maxCost* amount. Optional VAT number validation: 1. At any point in time before settlement the Charging Station may send a VatNumberValidationRequest to CSMS to verify a VAT number, that was entered by the EV Driver, to be included in the receipt. 2. CSMS verifies the VAT number using a local database or online service, and returns a VatNumberValidationResponse with *status* = `Accepted` and optional company name/address, when number is considered valid, or *status* = `Rejected` otherwise.

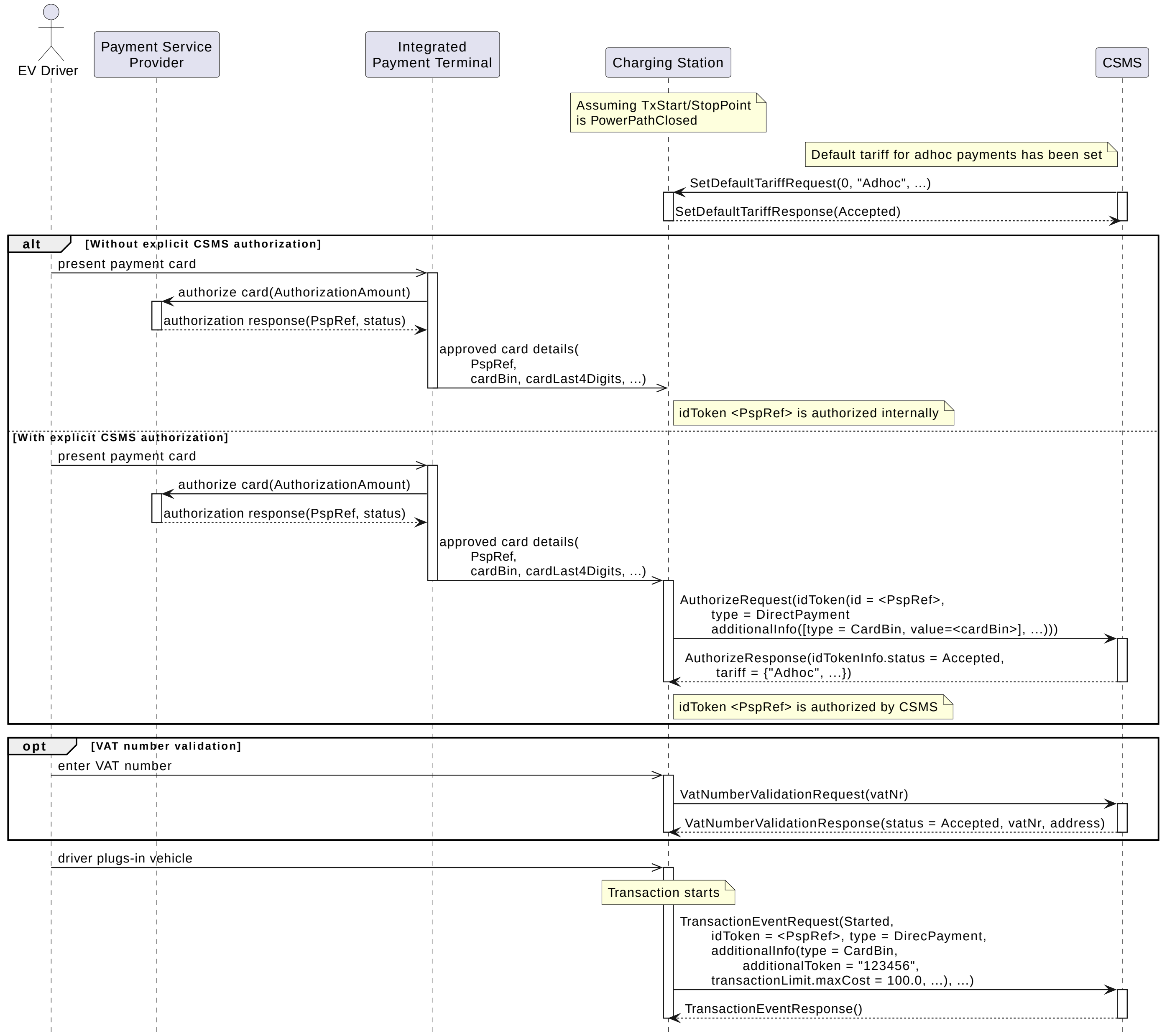

EV Driver
Payment Service Provider
Integrated Payment Terminal
Charging Station
CSMS
Assuming TxStart/StopPoint is PowerPathClosed
Default tariff for adhoc payments has been set
SetDefaultTariffRequest(0, "Adhoc", ...)
SetDefaultTariffResponse(Accepted)
alt [Without explicit CSMS authorization]
present payment card
authorize card(AuthorizationAmount)
authorization response(PspRef, status)
approved card details(PspRef, cardBin, cardLast4Digits, ...)
idToken <PspRef> is authorized internally
[With explicit CSMS authorization]
present payment card
authorize card(AuthorizationAmount)
authorization response(PspRef, status)
approved card details(PspRef, cardBin, cardLast4Digits, ...)
AuthorizeRequest(idToken(id = <PspRef>, type = DirectPayment additionalInfo([type = CardBin, value=<cardBin>], ...)))
AuthorizeResponse(idTokenInfo.status = Accepted, tariff = {"Adhoc", ...})
idToken <PspRef> is authorized by CSMS
opt [VAT number validation]
enter VAT number
VatNumberValidationRequest(vatNr)
VatNumberValidationResponse(status = Accepted, vatNr, address)
driver plugs-in vehicle
Transaction starts
TransactionEventRequest(Started, idToken = <PspRef>, type = DirecPayment, additionalInfo(type = CardBin, additionalToken = "123456", transactionLimit.maxCost = 100.0, ...), ...)
TransactionEventResponse()

Figure 39. Sequence diagram of an ad hoc authorization

C18 - Authorization using locally connected payment terminal - Requirements

ID	Precondition	Requirements	Note
C18.FR.01		Charging Station SHALL configure payment terminal to reserve the amount specified in configuration variable PaymentCtrlr.AuthorizationAmount when authorizing a payment card.	
C18.FR.02	When a payment card is presented	Charging Station SHALL instruct payment terminal to request authorization for this card from PSP.	Communication protocol between Charging Station and payment terminal and payment terminal and PSP is not part of OCPP.
C18.FR.03	When PSP authorizes the payment card AND PaymentCtrlr.AuthorizeDirectPayment is false	Charging Station SHALL authorize EV to charge.	

ID	Precondition	Requirements	Note
C18.FR.04	When PSP authorizes the payment card AND PaymentCtrlr.AuthorizeDirectPayment is true	Charging Station SHALL send an AuthorizeRequest with *idToken* = <PspRef>, *type* = `DirectPayment`, and *additionalInfo* containing a list of *types* configured in PaymentCtrlr.PaymentDetails and associated values in *additionalIdToken*.	Charging Station requests explicit authorization from CSMS. This can be useful if CSMS needs to be able to reject ad hoc payments or provide a non-default tariff plan.
C18.FR.05	C18.FR.04 AND CSMS gives authorization by responding with AuthorizeResponse with *status* = `Accepted`	Charging Station SHALL authorize EV to charge.	Authorization is still denied when *status* = `Accepted`, but EVSE is not on list of allowed EVSEs (see C01.FR.21)
C18.FR.06	C18.FR.03 OR C18.FR.05	The next TransactionEventRequest from Charging Station SHALL contain *idToken* = <PspRef>, *type* = `DirectPayment` and *additionalInfo* with a list of *types* and values in *additionalIdToken* as configured in PaymentCtrlr.PaymentDetails, and *transactionLimit.maxCost* set the to payment card authorization amount, as configured in PaymentCtrlr.AuthorizationAmount.	This is in accordance with E03.FR.01. Other fields then those mentioned here may also be relevant in the TransactionEventRequest message.
C18.FR.07	C18.FR.04 AND CSMS denies authorization by responding with AuthorizeResponse with *status* <> `Accepted`	Charging Station SHALL instruct payment terminal to release the authorization amount and cancel the payment.	Authorization is also denied when *status* = `Accepted`, but EVSE is not on list of allowed EVSEs (see C01.FR.21)
C18.FR.08		Charging Station MAY send a VatNumberValidationRequest to CSMS any time before settlement to validate a VAT number entered by EV Driver to be included on receipt.	VAT number may be required in certain legislations to be entered at time of sales.
C18.FR.09	C18.FR.08	CSMS validates given VAT number and responds with VatNumberValidationResponse with *status* = `Accepted` if validated and `Rejected` when invalid.	Optionally the company address can be returned in *company* field.
C18.FR.10	If VatNumberValidationRequest received by CSMS contains an *evseId*	CSMS SHALL use the same *evseId* in VatNumberValidationResponse.	This can make it simpler for Charging Station to associate the data with the correct transaction.

C19 - Cancelation prior to transaction

New in OCPP 2.1

No.	Type	Description
1	**Name**	Cancelation prior to transaction
2	**ID**	C19
3	**Objective**	To inform the CSMS that payment has been canceled and the authorization amount has been released.
4	**Description**	The EV Driver cancels the payment prior to starting an actual OCPP transaction, or the EVConnectionTimeout occurs.
	Actors	EV Driver, Payment Terminal, CSMS, Charging Station
	Scenario description	*PaymentCtrlr.AuthorizeDirectPayment = false* 1. Payment card of EV Driver has been authorized by the payment service provider and an authorization amount has been reserved for the card. 2. Charging Station has authorized *idToken* for charging. 3. EV Driver decides to cancel the session **before** the EV is plugged-in, or EV Driver does not plug in EV and a EVConnectionTimeout occurs. 4. Charging Station notifies payment terminal about cancelation. 5. Payment terminal requests payment service provider to release the authorization reservation on the card. 6. Charging Station ends authorization of *idToken*.

No.	Type	Description
	Alternative scenario(s)	*PaymentCtrlr.AuthorizeDirectPayment = true* 1. Payment card of EV Driver has been authorized by the payment service provider and an authorization amount has been reserved on the card. 2. Charging Station sends an AuthorizeRequest for *idToken* = <PspRef>, *type* = `DirectPayment` and associated *additionalInfo*. 3. CSMS responds with AuthorizeResponse to authorize *idToken* for charging. 4. EV Driver decides to cancel the session **before** the EV is plugged-in, or EV Driver does not plug in EV and a EVConnectionTimeout occurs. 5. Payment terminal requests payment service provider to release the authorization reservation on the card. 6. Charging Station ends authorization of *idToken*. 7. Charging Station sends a NotifySettlementRequest with *pspRef* = *idToken* and *status* = `Canceled` to notify CSMS that the previously authorized *idToken* will not start a transaction and payment has been canceled.
5	**Prerequisites**	The EV Driver has used the payment terminal to authorize. No TransactionEventRequest has yet been sent, because TxStartPoint = "PowerPathClosed" or "EVConnected" and EV is not plugged-in.
6	**Postcondition(s)**	The payment transaction has been canceled.
7	**Error handling**	
8	**Remark(s)**	If PaymentCtrlr.AuthorizeDirectPayment is false, then CSMS is not notified of the payment canceling, because it will not be aware of the *idToken*. See use case C20 for cancelation when a transaction has already been started.

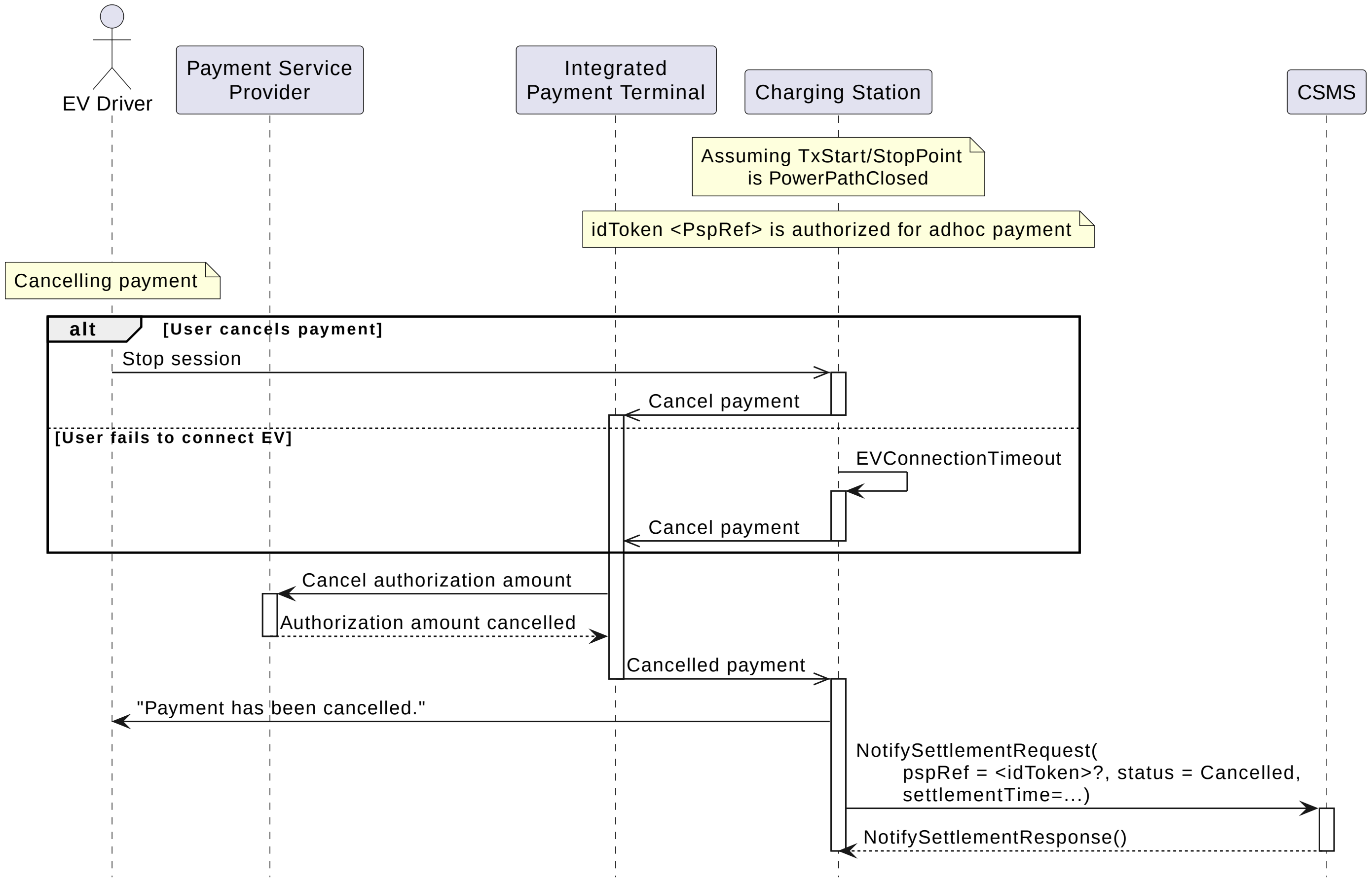

EV Driver
Payment Service Provider
Integrated Payment Terminal
Charging Station
CSMS
Assuming TxStart/StopPoint is PowerPathClosed
idToken <PspRef> is authorized for adhoc payment
Cancelling payment
alt
[User cancels payment]
Stop session
Cancel payment
[User fails to connect EV]
EVConnectionTimeout
Cancel payment
Cancel authorization amount
Authorization amount cancelled
Cancelled payment
"Payment has been cancelled."
NotifySettlementRequest(pspRef = <idToken>?, status = Cancelled, settlementTime=...)
NotifySettlementResponse()

Figure 40. Sequence diagram of canceling an ad hoc authorization

C19 - Cancelation prior to transaction - Requirements

ID	Precondition	Requirements	Note
C19.FR.01	When EV Driver stops the session at Charging Station before a transaction has been started OR An EVConnectionTimeout occurs	Charging Station SHALL instruct payment terminal to release the authorization amount and cancel the payment.	
C19.FR.02	C19.FR.01 AND PaymentCtrlr.AuthorizeDirectPayment = true	Charging Station SHALL send a NotifySettlementRequest with *idToken* = <pspRef> of the ad hoc payment and *status* = `Canceled` and `settlementTime` set to current time.	This notifies CSMS that the DirectPayment *idToken* will not be charged.

C20 - Cancelation after start of transaction

New in OCPP 2.1

No.	Type	Description
1	**Name**	Cancelation after start of transaction
2	**ID**	C20
3	**Objective**	To inform the CSMS that payment has been canceled and the authorization amount has been released.
4	**Description**	The EV Driver cancels the payment when a transaction has been started, but no energy has been delivered and no other costs have been made (e.g. no reservation fee).
	Actors	EV Driver, Payment Terminal, CSMS, Charging Station
	Scenario description	*TxStartPoint = "PowerPathClosed" and charging does not start* 1. Payment card of EV Driver has been authorized by the payment service provider and an authorization amount has been reserved on the card. 2. Charging Station has authorized *idToken* for charging. 3. EV Driver connects EV. 4. Charging Station sends a TransactionEventRequest with *eventType* = `Started`, *triggerReason* = `CablePluggedIn` and *idToken* information and other relevant fields. a. CSMS responds with TransactionEventResponse with *idTokenInfo*. 5. Charging does not start, e.g. due to failure in either EV or Charging Station. 6. EV Driver decides to cancel the session. 7. Payment terminal requests payment service provider to release the authorization reservation on the card. 8. Charging Station ends authorization of *idToken*. 9. Charging Station sends TransactionEventRequest with *eventType* = `Ended` and *triggerReason* = `StopAuthorized` and other relevant fields. 10. CSMS responds with TransactionEventResponse with *totalCost* = 0. 11. Charging Station sends a NotifySettlementRequest with *pspRef* = *idToken* and *status* = `Canceled` to notify CSMS that the previously authorized *idToken* will not start a transaction and payment has been canceled.

No.	Type	Description
	Alternative scenario	*TxStartPoint = "Authorized" and not plugged-in* 1. Payment card of EV Driver has been authorized by the payment service provider and an authorization amount has been reserved on the card. 2. Charging Station has authorized *idToken* for charging. 3. Charging Station sends a TransactionEventRequest with *eventType* = `Started` and *idToken* information and other relevant fields. a. CSMS responds with TransactionEventResponse with *idTokenInfo*. 4. EV Driver does not plug in EV and a EVConnectionTimeout occurs. 5. Payment terminal requests payment service provider to release the authorization reservation on the card. 6. Charging Station ends authorization of *idToken*. 7. Charging Station sends TransactionEventRequest with *eventType* = `Ended` and *triggerReason* = `EVConnectTimeout` and other relevant fields. 8. CSMS responds with TransactionEventResponse with *totalCost* = 0. 9. Charging Station sends a NotifySettlementRequest with *pspRef* = *idToken* and *status* = `Canceled` to notify CSMS that the previously authorized *idToken* will not start a transaction and payment has been canceled.
5	**Prerequisites**	The EV Driver has used the payment terminal to authorize. A transaction was started, but no costs have been incurred yet.
6	**Postcondition(s)**	The payment transaction has been canceled and transaction has been ended.
7	**Error handling**	
8	**Remark(s)**	See use case C19 for cancelation when a transaction has not yet been started.

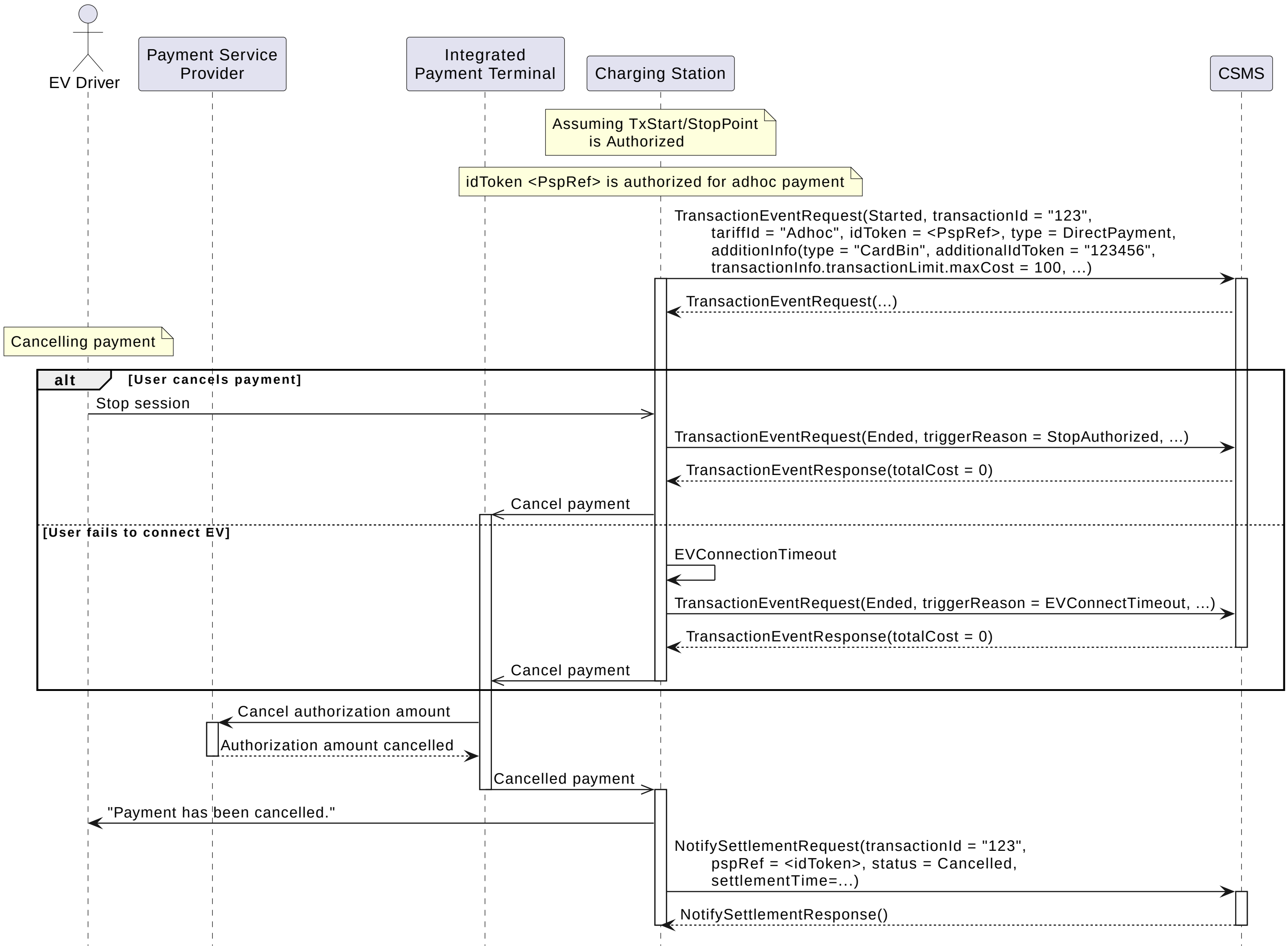

EV Driver
Payment Service Provider
Integrated Payment Terminal
Charging Station
CSMS
Assuming TxStart/StopPoint is Authorized
idToken <PspRef> is authorized for adhoc payment
TransactionEventRequest(Started, transactionId = "123", tariffId = "Adhoc", idToken = <PspRef>, type = DirectPayment, additionInfo(type = "CardBin", additionalIdToken = "123456", transactionInfo.transactionLimit.maxCost = 100, ...)
TransactionEventRequest(...)
Cancelling payment
alt
[User cancels payment]
Stop session
TransactionEventRequest(Ended, triggerReason = StopAuthorized, ...)
TransactionEventResponse(totalCost = 0)
Cancel payment
[User fails to connect EV]
EVConnectionTimeout
TransactionEventRequest(Ended, triggerReason = EVConnectTimeout, ...)
TransactionEventResponse(totalCost = 0)
Cancel payment
Cancel authorization amount
Authorization amount cancelled
Cancelled payment
"Payment has been cancelled."
NotifySettlementRequest(transactionId = "123", pspRef = <idToken>, status = Cancelled, settlementTime=...)
NotifySettlementResponse()

Figure 41. Sequence diagram of canceling an ad hoc authorization after start of transaction

C20 - Cancelation after start of transaction - Requirements

ID	Precondition	Requirements	Note
C20.FR.01	When EV Driver stops the session at Charging Station after a transaction has been started	Charging Station SHALL send a TransactionEventRequest with *eventType* = `Ended` and *triggerReason* = `StopAuthorized` and any other relevant fields.	
C20.FR.02	When EV Driver does not connect EV in time after a transaction has been started, and an EVConnectionTimeout occurs	Charging Station SHALL send a TransactionEventRequest with *eventType* = `Ended` and *triggerReason* = `EVConnectTimeout`	
C20.FR.03	C20.FR.01 OR C20.FR.02	CSMS SHALL respond with a TransactionEventResponse with *totalCost* = 0	Prerequisite of this use case is that no costs have been incurred.
C20.FR.04	C20.FR.03	Charging Station SHALL instruct payment terminal to release the authorization amount and cancel the payment.	
C20.FR.05	C20.FR.04	Charging Station SHALL send a NotifySettlementRequest with *transactionId* of transaction, *idToken* of the ad hoc payment in *pspRef*, and *status* = `Canceled` and `settlementTime` set to current time.	This notifies CSMS that the DirectPayment *idToken* has not been charged for the transaction.

C21 - Settlement at end of transaction

New in OCPP 2.1

No.	Type	Description
1	**Name**	Settlement at end of transaction
2	**ID**	C21
3	**Objective**	Settle the amount of charging session and generate receipt for the EV Driver.
4	**Description**	When the transaction ends, the EV Driver is shown the total cost. The amount that is being settled, and the successful settlement status and time are sent to the CSMS and a receipt is provided.
	Actors	EV Driver, Payment Terminal, CSMS, Charging Station
	Scenario description	*URL to receipt is provided by CSMS: PaymentCtrlr.ReceiptByCSMS = true, SettlementByCSMS = false* 1. When transaction ends, Charging Station sends a TransactionEventRequest with *eventType* = `Ended` and *costDetails* to CSMS. 2. Charging Station shows the cost breakdown in *costDetails* to EV Driver. 3. Charging Station instructs payment terminal to settle the payment for the amount in *costDetails.totalCost*. 4. When payment is settled, Charging Station sends a NotifySettlementRequest with *transactionId* of the transaction, *pspRef* with value of *idToken*, *status* = `Settled`, *settlementAmount* and *settlementTime* and optionally *vatNumber*, *vatCompany*. a. CSMS responds with a NotifySettlementResponse with *receiptUrl*. b. Charging Station displays *receiptUrl* from CSMS to EV Driver.

No.	Type	Description
	Alternative scenario	*URL to receipt is provided by payment terminal: PaymentCtrlr.ReceiptByCSMS = false, SettlementByCSMS = false* 1. When transaction ends, Charging Station sends a TransactionEventRequest with *eventType* = `Ended` and *costDetails* to CSMS. 2. Charging Station shows the cost breakdown in *costDetails* to EV Driver. 3. Charging Station instructs payment terminal to settle the payment for the amount in *costDetails.totalCost*. 4. When payment is settled, payment terminal sends the URL that points to the receipt. 5. Charging Station sends a NotifySettlementRequest with *transactionId* of the transaction, *pspRef* with value of *idToken*, *status* = `Settled`, *settlementAmount* and *settlementTime*` and the receipt information from payment service provder: *receiptID*, *receiptUrl*. a. CSMS responds with a NotifySettlementResponse. b. Charging Station displays *receiptUrl* from payment terminal to EV Driver.
	Alternative scenario	*Settlement by CSMS: SettlementByCSMS = true* 1. When transaction ends, Charging Station sends a TransactionEventRequest with *eventType* = `Ended` and *costDetails* to CSMS. 2. Charging Station shows the cost breakdown in *costDetails* to EV Driver. 3. Configuration variable PaymentCtrlr.SettlementByCSMS = true, therefore settlement is performed directly with PSP, bypassing the payment terminal. a. CSMS instructs payment service provider to settle the payment for the amount in *costDetails.totalCost*. **NOTE** In this scenario there is no feedback to EV Driver to provide a receipt. The CSO will need to offer the ability to retrieve a receipt via a website or app.
5	**Prerequisites**	Transaction was started using ad hoc payment. Charging Station supports local cost calculation.
6	**Postcondition(s)**	CSMS knows of the settled status and the EV Driver has been shown the cost and means to retrieve a receipt.
7	**Error handling**	
8	**Remark(s)**	A receipt URL, e.g. as a QR code, might be shown on the payment terminal display. Showing the receipt URL after settlement is problematic if the TxStopPoint = ParkingBayOccupancy, because the EV Driver has left already at that moment. See chapter I. Tariff and Cost for details about the cost breakdown.

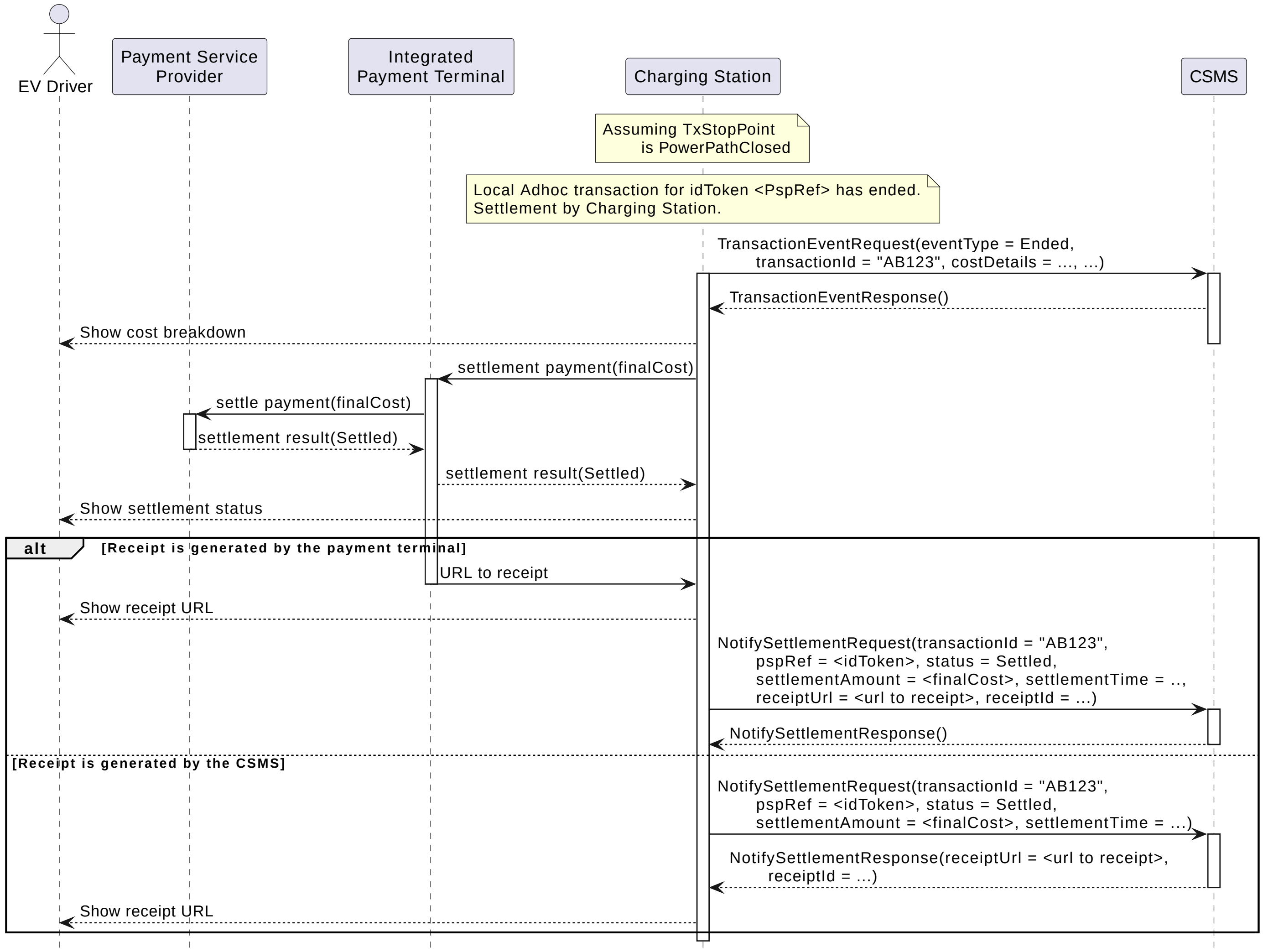

EV Driver
Payment Service Provider
Integrated Payment Terminal
Charging Station
CSMS
Assuming TxStopPoint is PowerPathClosed
Local Adhoc transaction for idToken <PspRef> has ended. Settlement by Charging Station.
TransactionEventRequest(eventType = Ended, transactionId = "AB123", costDetails = ..., ...)
TransactionEventResponse()
Show cost breakdown
settlement payment(finalCost)
settle payment(finalCost)
settlement result(Settled)
settlement result(Settled)
Show settlement status
alt
[Receipt is generated by the payment terminal]
URL to receipt
Show receipt URL
NotifySettlementRequest(transactionId = "AB123", pspRef = <idToken>, status = Settled, settlementAmount = <finalCost>, settlementTime = .., receiptUrl = <url to receipt>, receiptId = ...)
NotifySettlementResponse()
[Receipt is generated by the CSMS]
NotifySettlementRequest(transactionId = "AB123", pspRef = <idToken>, status = Settled, settlementAmount = <finalCost>, settlementTime = ...)
NotifySettlementResponse(receiptUrl = <url to receipt>, receiptId = ...)
Show receipt URL

Figure 42. Sequence diagram of payment settlement by Charging Station

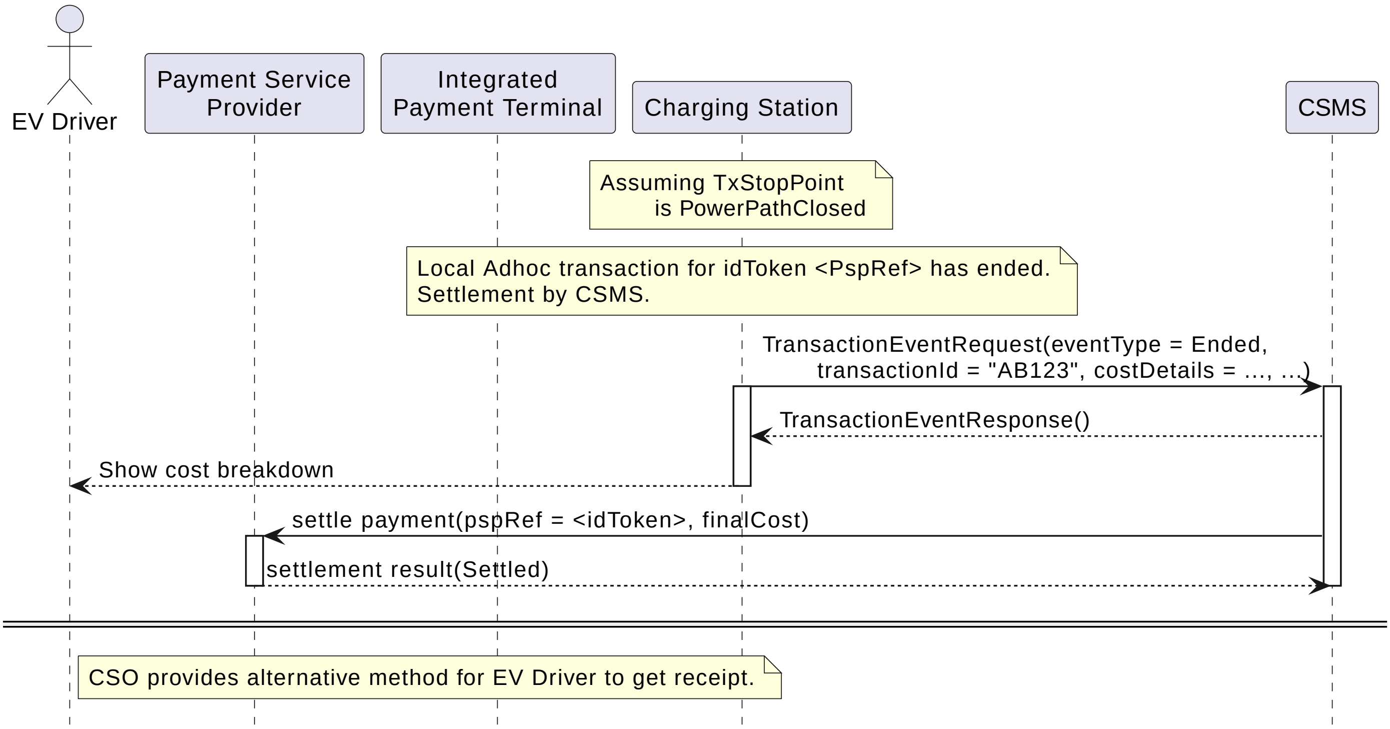

EV Driver
Payment Service Provider
Integrated Payment Terminal
Charging Station
CSMS
Assuming TxStopPoint is PowerPathClosed
Local Adhoc transaction for idToken <PspRef> has ended. Settlement by CSMS.
TransactionEventRequest(eventType = Ended, transactionId = "AB123", costDetails = ..., ...)
TransactionEventResponse()
Show cost breakdown
settle payment(pspRef = <idToken>, finalCost)
settlement result(Settled)
CSO provides alternative method for EV Driver to get receipt.

Figure 43. Sequence diagram of payment settlement by CSMS

C21 - Settlement at end of transaction - Requirements

ID	Precondition	Requirements	Note
C21.FR.01	When a locally started transaction ends that was authorized by an *idToken* of *type* = `DirectPayment` AND PaymentCtrlr.SettlementByCSMS is absent or false	Charging Station SHALL settle the total cost of the transaction via the payment terminal.	
C21.FR.02	When Charging Station receives the settlement status from payment terminal	Charging Station SHALL send a NotifySettlementRequest to CSMS with *status*, *settlementAmount*, *settlementTime*, *transactionId* and *pspRef*.	Optionally a *vatNumber* and *vatCompany* can be added.
C21.FR.03	C21.FR.02 AND PaymentCtrlr.ReceiptByCSMS = true	CSMS SHALL respond with NotifySettlementResponse with *receiptUrl*.	
C21.FR.04	C21.FR.02 AND PaymentCtrlr.ReceiptByCSMS = false	Charging Station SHALL add the URL to the receipt in the NotifySettlementRequest in field *receiptUrl* and the ID of the receipt in *receiptId*.	
C21.FR.05	When a locally started transaction ends that was authorized by an *idToken* of *type* = `DirectPayment` AND PaymentCtrlr.SettlementByCSMS is true	CSMS SHALL settle the total cost of the transaction with the payment service provider.	The PSP reference to settle the payment session is in *idToken*.
C21.FR.06	C21.FR.05	Charging Station SHALL NOT settle the total cost of the transaction via the payment terminal.	
C21.FR.07	C21.FR.05	Charging Station MAY display the value of PaymentCtrlr.ReceiptServerUrl to the EV Driver as a location where a receipt can be retrieved at a later time.	This information can, for example, be shown immediately after authorization and/or when stopping the transaction.

C22 - Settlement is rejected or fails

New in OCPP 2.1

No.	Type	Description
1	**Name**	Settlement is rejected or fails
2	**ID**	C22
3	**Objective**	To inform the CSMS that the transaction settlement has been rejected or otherwise failed to be successfully settled.
4	**Description**	The Charging Station failed to settle the amount. Settlement can be rejected by the payment service provider, or it can fail for technical reasons.
	Actors	EV Driver, Payment Terminal, CSMS, Charging Station
	Scenario description	*PSP rejects settlement* 1. Charging Station requests payment terminal to settle payment. 2. PSP rejects settlement. 3. Payment terminal returns a rejection and error code. 4. Charging Station sends a NotifySettlementRequest with *transactionId*, *idToken*, *settlementAmount*, *settlementTime* and *status* = `Rejected` and (optional) additional error information in *statusInfo*.

No.	Type	Description
	Alternative scenario	*Settlement fails due to communication problems* 1. Charging Station requests payment terminal to settle payment. 2. Payment terminal attempts to contact PSP, but fails. 3. Payment terminal may be configured to autonomously perform a number of retries. (This is not an OCPP-related configuration). 4. Payment terminal returns a Failed status 5. Charging Station sends a NotifySettlementRequest with *transactionId*, *idToken*, *settlementAmount*, *settlementTime* and *status* = `Failed` and (optional) additional error information in *statusInfo*.
5	**Prerequisites**	Transaction was started using ad hoc payment.
6	**Postcondition(s)**	CSMS knows of the rejected/failed settled status.
7	**Error handling**	
8	**Remark(s)**	In case of settlement failure CSO can try to capture the amount by contacting the payment service provider and providing the payment details, such as the PSP reference (which is the value of *idToken*).

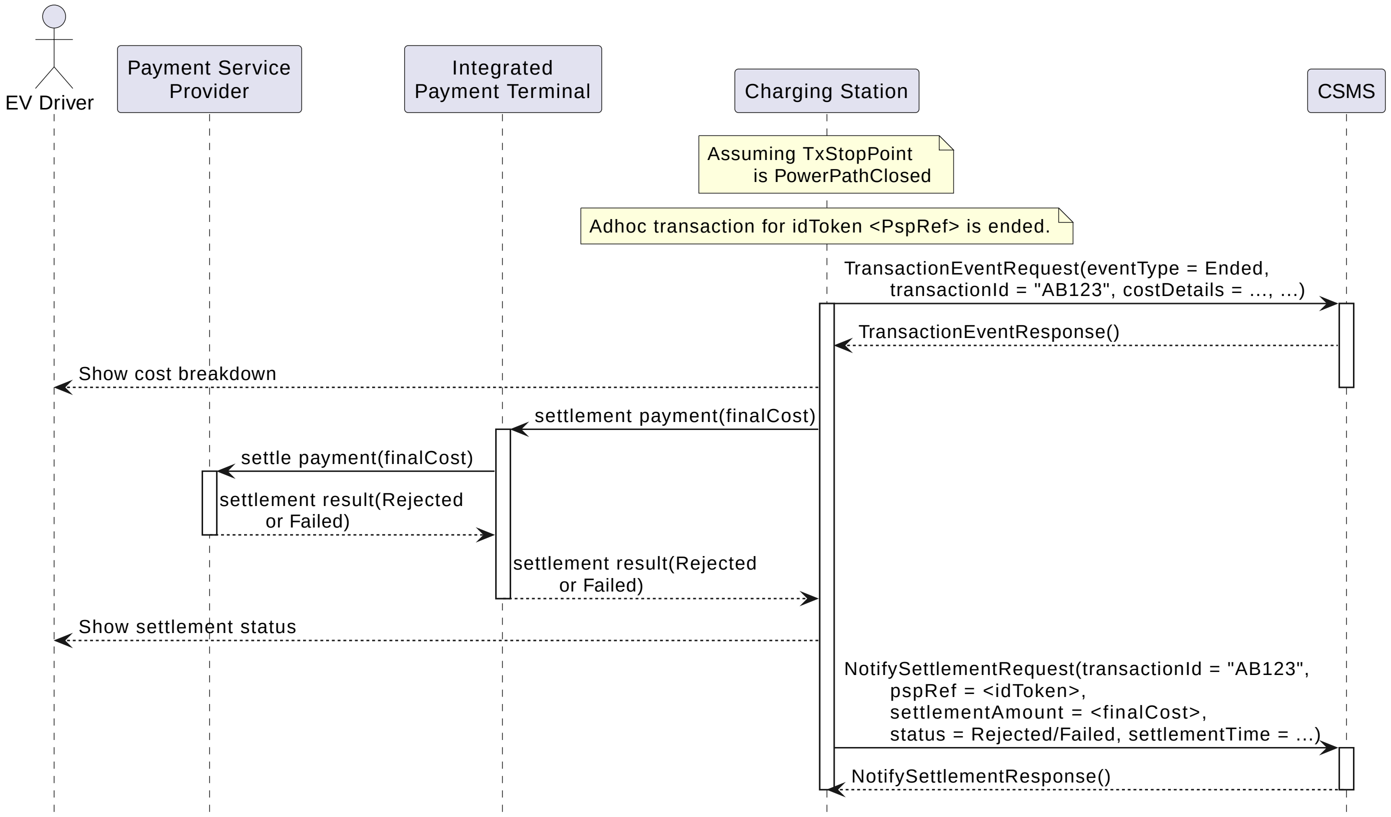

EV Driver
Payment Service Provider
Integrated Payment Terminal
Charging Station
CSMS
Assuming TxStopPoint is PowerPathClosed
Adhoc transaction for idToken <PspRef> is ended.
TransactionEventRequest(eventType = Ended, transactionId = "AB123", costDetails = ..., ...)
TransactionEventResponse()
Show cost breakdown
settlement payment(finalCost)
settle payment(finalCost)
settlement result(Rejected or Failed)
settlement result(Rejected or Failed)
Show settlement status
NotifySettlementRequest(transactionId = "AB123", pspRef = <idToken>, settlementAmount = <finalCost>, status = Rejected/Failed, settlementTime = ...)
NotifySettlementResponse()

Figure 44. Sequence diagram of failing settlement of an ad hoc payment

C22 - Settlement is rejected or fails - Requirements

ID	Precondition	Requirements	Note
C22.FR.01	When Charging Station receives a settlement status from payment terminal that is `Rejected`	Charging Station SHALL send a NotifySettlementRequest to CSMS with *status* = `Rejected`, *settlementAmount*, *settlementTime*, *transactionId* and *pspRef* without receipt information.	This means the payment service provider has rejected the settlement.
C22.FR.02	When Charging Station receives a settlement status from payment terminal that is `Failed`	Charging Station SHALL send a NotifySettlementRequest to CSMS with *status* = `Failed`, *settlementAmount*, *settlementTime*, *transactionId* and *pspRef* without receipt information.	This means that there are technical reasons why the settlement failed, e.g. a failure to communicate.

C23 - Increasing authorization amount

New in OCPP 2.1

No.	Type	Description
1	**Name**	Increasing authorization amount
2	**ID**	C23
3	**Objective**	To increase the authorization amount during the transaction when costs are exceeding the original authorization amount.
4	**Description**	Costs of the transaction are exceeding the authorization amount that was reserved when the payment session started. If it is supported by the payment terminal, the authorization amount can be increased. Support for incremental authorization by the payment terminal is indicated by PaymentCtrlr.IncrementalAuthorizationAmount existing with a value > 0.
	Actors	EV Driver, Payment Terminal, CSMS, Charging Station
	Scenario description	1. A transaction with an ad hoc payment is active. 2. Payment terminal initially sets an authorization amount of PaymentCtrlr.AuthorizationAmount. 3. When PaymentCtrlr.IncrementalAuthorizationAmount > 0 and transaction costs exceed the current authorization amount minus PaymentCtrlr.IncrementalAuthorizationThreshold a. Charging Station instructs payment terminal to increase authorization with PaymentCtrlr.IncrementalAuthorizationAmount. b. Charging Station increases the transaction limit in *transactionInfo.maxCost* with the same amount.
5	**Prerequisites**	Transaction was started using ad hoc payment and PaymentCtrlr.IncrementalAuthorizationAmount exists and has a value > 0
6	**Postcondition(s)**	The authorization amount and transactionInfo.maxCost has been increased.
7	**Error handling**	
8	**Remark(s)**	If the maximum authorization amount is reached and PaymentCtrlr.IncrementalAuthorizationAmount = 0 (or absent), then the energy flow will be halted, because *transactionInfo.maxCost* is reached. The authorization amount needs to be increased before is has been depleted in order to avoid interruption of the energy flow. The moment when to increase the amount is determined by PaymentCtrlr.IncrementalAuthorizationThreshold.

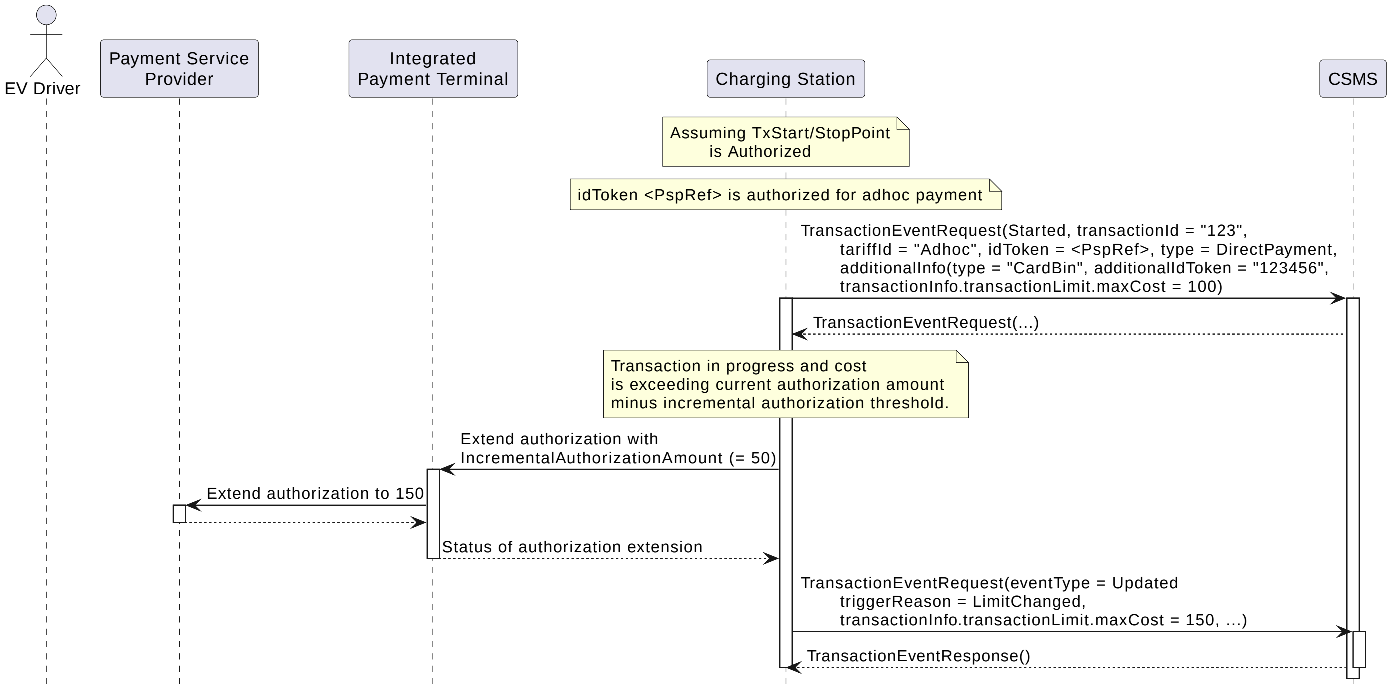

EV Driver
Payment Service Provider
Integrated Payment Terminal
Charging Station
CSMS
Assuming TxStart/StopPoint is Authorized
idToken <PspRef> is authorized for adhoc payment
TransactionEventRequest(Started, transactionId = "123", tariffId = "Adhoc", idToken = <PspRef>, type = DirectPayment, additionalInfo(type = "CardBin", additionalIdToken = "123456", transactionInfo.transactionLimit.maxCost = 100)
TransactionEventRequest(...)
Transaction in progress and cost is exceeding current authorization amount minus incremental authorization threshold.
Extend authorization with IncrementalAuthorizationAmount (= 50)
Extend authorization to 150
Status of authorization extension
TransactionEventRequest(eventType = Updated triggerReason = LimitChanged, transactionInfo.transactionLimit.maxCost = 150, ...)
TransactionEventResponse()

Figure 45. Sequence diagram showing incremental authorization

C23 - Incremental authorization - Requirements

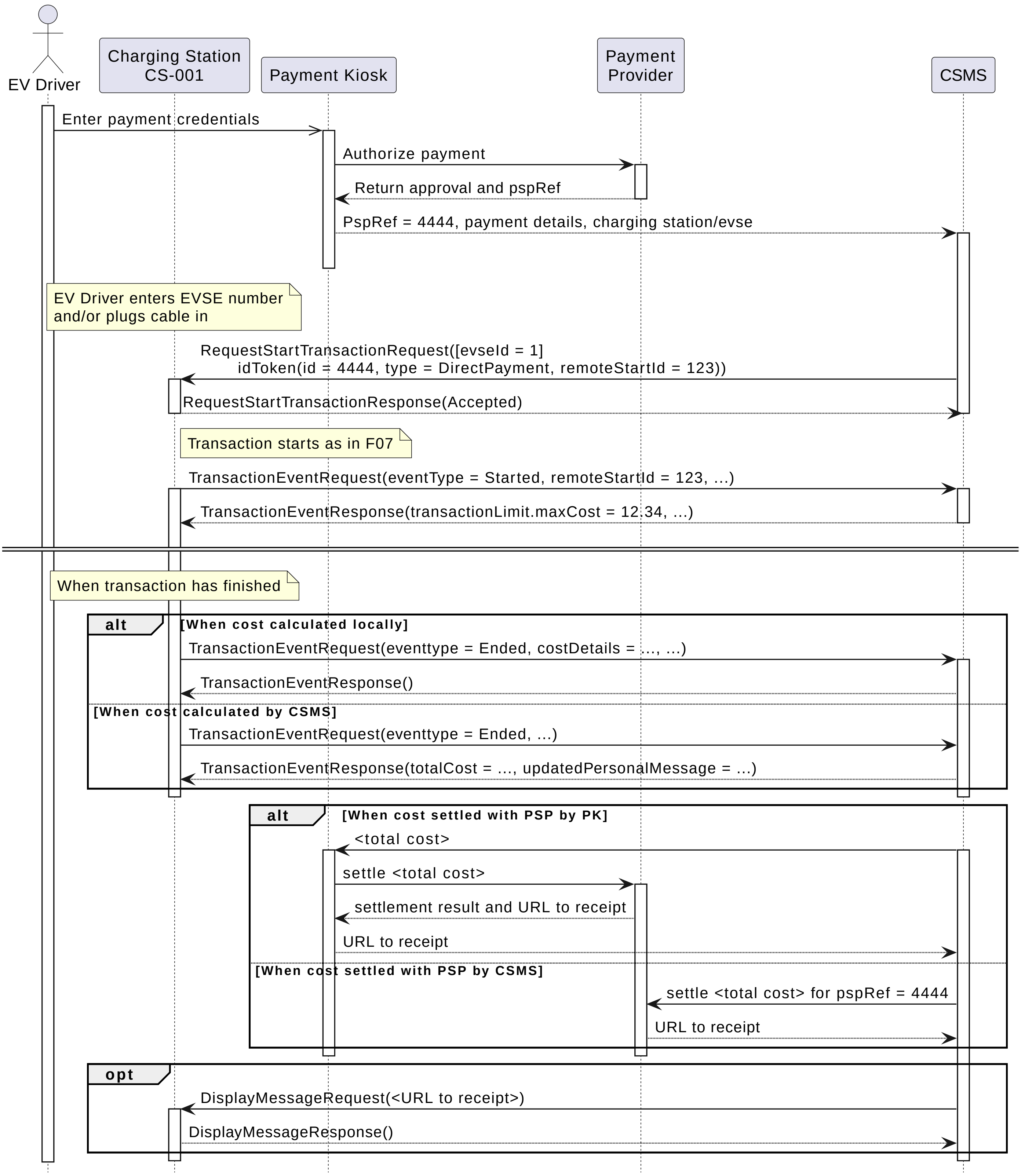
ID	Precondition	Requirements	Note
C23.FR.01	When configuration variable PaymentCtrlr.IncrementalAuthorizationAmount exists and has a value > 0	Charging Station SHALL request payment terminal to increase the authorization amount when total transaction costs exceed the current authorization amount minus PaymentCtrlr.IncrementalAuthorizationThreshold.	When PaymentCtrlr.IncrementalAuthorizationThreshold is absent this threshold is defined by the implementation.
C23.FR.02	C23.FR.01 AND Increasing the authorization with PaymentCtrlr.IncrementalAuthorizationAmount was successful	Charging Station SHALL send a TransactionEventRequest with *triggerReason* = `LimitSet` and a *transactionLimit.maxCost* that is increased by PaymentCtrlr.IncrementalAuthorizationAmount.	

C24 - Ad hoc payment via stand-alone payment terminal

New in OCPP 2.1

No.	Type	Description
1	**Name**	Ad hoc payment via stand-alone payment terminal
2	**ID**	C24
3	**Objective**	To facilitate payment with a credit/debit card via a stand-alone payment terminal/kiosk (i.e. not integrated into the charging station).
4	**Description**	A Charging Station (or a group of Charging Stations) has a central payment terminal/kiosk. An EV Driver uses a credit/debit card to pay for charging. The transaction is authorized by the PSP, the CSMS receives a message from the PSP, and remotely starts the transaction on the Charging Station.
	Actors	EV Driver, Payment Kiosk, Payment Service Provider, CSMS, Charging Station
	Scenario description	*Authorizing payment card* 1. EV Driver presents payment card at payment kiosk/terminal (PK) to initiate a charging session. 2. PK communicates with the Payment Service Provider (PSP). 3. Upon approval PSP returns an approved status to PK and a unique reference for this payment in the PSP platform. a. This unique reference is called "pspRef" in OCPP. 4. PK sends charging station/evse and payment details, such as pspRef, cardBin, cardLast4Digits, etc. to CSMS. 5. CSMS sends a RequestStartTransactionRequest (optionally with *evseId*) to Charging Station with *idToken* having value <pspRef> and *type* = `DirectPayment`. 6. Charging Station continues as per use cases F07 "Remote start with fixed cost". *(User input may be required at Charging Station to select the EVSE to use.)*

No.	Type	Description
	Scenario description	*Settlement when transaction has finished* 1. If cost calculated locally a. Charging Station sends a TransactionEventRequest with *eventType* = Ended and *costDetails* to CSMS. b. CSMS responds with TransactionEventResponse. c. Charging Station shows the cost breakdown in *costDetails* to EV Driver. 2. If cost calculated centrally a. Charging Station sends a TransactionEventRequest with *eventType* = Ended to CSMS. b. CSMS responds with TransactionEventResponse with *totalCost* and an *updatedPersonalMessage*. c. Charging Station shows the cost breakdown in *updatedPersonalMessage* to EV Driver. 3. If settlement with PSP via PK a. CSMS sends total cost to PK for settling with PSP b. PSP sends receipt data or a URL to a receipt to PK. c. PK returns receipt data or a URL to the receipt to CSMS. 4. If settlement with PSP via CSMS a. CSMS performs settlement directly with PSP based on the <pspRef>. b. PSP returns a receipt data or a URL to a receipt to CSMS. 5. CSMS may use SetDisplayMessageRequest to display a URL or QR code to a location where the EV Driver can retrieve a receipt.
5	**Prerequisites**	PK is able to communicate with CSMS.
6	**Postcondition(s)**	CSMS remotely starts a transaction for EV Driver.
7	**Error handling**	
8	**Remark(s)**	This was use case C03 in OCPP 2.0.1. The communication between CSMS and PK is out of scope of this document.


EV Driver
Charging Station CS-001
Payment Kiosk
Payment Provider
CSMS
Enter payment credentials
Authorize payment
Return approval and pspRef
PspRef = 4444, payment details, charging station/evse
EV Driver enters EVSE number and/or plugs cable in
RequestStartTransactionRequest([evseId = 1] idToken(id = 4444, type = DirectPayment, remoteStartId = 123))
RequestStartTransactionResponse(Accepted)
Transaction starts as in F07
TransactionEventRequest(eventType = Started, remoteStartId = 123, ...)
TransactionEventResponse(transactionLimit.maxCost = 12.34, ...)
When transaction has finished
alt
[When cost calculated locally]
TransactionEventRequest(eventtype = Ended, costDetails = ..., ...)
TransactionEventResponse()
[When cost calculated by CSMS]
TransactionEventRequest(eventtype = Ended, ...)
TransactionEventResponse(totalCost = ..., updatedPersonalMessage = ...)
alt
[When cost settled with PSP by PK]
<total cost>
settle <total cost>
settlement result and URL to receipt
URL to receipt
[When cost settled with PSP by CSMS]
settle <total cost> for pspRef = 4444
URL to receipt
opt
DisplayMessageRequest(<URL to receipt>)
DisplayMessageResponse()

C24 - Ad hoc payment via stand-alone payment terminal - Requirements

ID	Precondition	Requirements	Note
C24.FR.01	When CSMS receives a message from Payment Kiosk that payment is authorized with authorization reference <PspRef> for Charging Station with identity <CS> and optionally an EVSE <evse>	CSMS SHALL send a RequestStartTransactionRequest to Charging Station with identity <CS> and EVSE <evse> (when present) and having *idToken.idToken* = <PspRef> and *idToken.type* = `DirectPayment`	There is need to include *additionalInfo* elements for payment details, like cardBin, cardLast4Digits, towards Charging Station. The protocol between Payment Kiosk and CSMS is not specified in this document.

ID	Precondition	Requirements	Note
C24.FR.02	C24.FR.01 AND Charging Station sends a TransactionEventRequest with *eventType* = `Started`	CSMS SHALL respond with a TransactionEventResponse with *transactionLimit* set to the required limit.	*transactionLimit* uses *maxCost* for cost limit, *maxEnergy* for energy limit, *maxTime* for time limit or *maxSoC* for SoC limit.
C24.FR.03	C24.FR.02	Charging Station SHALL include this limit in the field *transactionInfo.transactionLimit* (once) in the next TransactionEventRequest	
C24.FR.04		Charging Station SHALL behave as in use case E16 - Transactions with fixed cost, energy, SoC or time.	
Settlement			
C24.FR.10	After CSMS received a TransactionEventRequest with *eventType* = Ended or the transaction	CSMS SHALL send a message to Payment Kiosk or PSP to settle the cost	Whether settling is done via Payment Kiosk or directly with PSP is depending on the implementation. Cost may have been calculated locally or centrally.
C24.FR.11	When CSMS receives receipt information or a URL to a receipt from Payment Kiosk or PSP	CSMS MAY send a SetDisplayMessageRequest with a URL where EV Driver can download the receipt .	Other methods to find and download receipts may also be offered by CSO.

C25 - Ad hoc payment via a QR code

New in OCPP 2.1

No.	Type	Description
1	**Name**	Ad hoc payment via a QR code
2	**ID**	C25
3	**Objective**	To provide a static or dynamic QR code with a URL for ad hoc payment
4	**Description**	Static QR code: The Charging Station has a static QR code (often a sticker) to scan to start a transaction. Dynamic QR code: The Charging Station displays a dynamic QR code with a time-based one-time password that is based on a shared secret between Charging Station and CSMS. The EV Driver scans the QR code to access a website where the EV Driver can enter payment credentials for the charging service. In case of a dynamic QR code the CSMS checks that the URL scanned from the QR code is authentic. Upon successful payment authorization CSMS issues a remote start of a transaction on the Charging Station (and optional EVSE) that is encoded in the URL.
	Actors	EV Driver, PSP, CSMS, Charging Station

No.	Type	Description
	Scenario description #1	*Scanning a dynamic QR code* 1. Charging Station displays a QR code with a time-based one-time password. a. Optionally, Charging Station requests input from EV Driver to specify a maximum time, energy or cost prior to displaying the QR code b. The input is added as query parameters to the URL in the QR code. 2. EV Driver scans QR code with URL to access a web page of CSMS to initiate a charging session. 3. CSMS decodes URL according to the configured URL template and validates the time-based one-time password. a. The URL contains as a minimum the Charging Station identity, one time password and version, and optionally the EVSE number, but other parameters can be configured in the URL template as well. b. If URL validation or decoding fails no charging session will be started. *Use case ends here.* 4. CSMS optionally sends a NotifyWebPaymentStartedRequest message with *evseId* and *timeout* to notify Charging Station of the event. a. Charging Station displays feedback to EV Driver and prevents that a transaction is started locally on the EVSE in *evseId* for as long as *timeout* seconds, or until the RequestStartTransactionRequest from CSMS is received. 5. CSMS diverts EV Driver to website of PSP (Payment Service Provider). 6. EV Driver enters payment credentials in web page shown by PSP. 7. Upon approval PSP returns an approved status and a unique reference for this payment in the PSP platform. a. This unique reference is called "pspRef" in OCPP. b. PSP sends payment details, such as PspRef, cardBin, cardLast4Digits, etc. to CSMS. 8. CSMS sends a RequestStartTransactionRequest to Charging Station and EVSE from the URL and with *idToken* having value <pspRef> and *type* = `DirectPayment` . 9. Use case continues as per use case "F07 Remote start with cost limit" in which CSMS will provide a *transactionLimit.maxCost* set to the payment card authorization amount, which acts as a ceiling for the cost, independent of a value entered bij EV Driver in step 1.
	Scenario description #2	*Scanning a static QR code* **WARNING** A static QR code is susceptible to fraud. A new QR sticker that points to a fake payment site can be pasted over the original QR sticker. 1. Charging Station displays a static QR code (often a sticker). 2. EV Driver scans QR code with URL to access a web page of CSMS to initiate a charging session. 3. CSMS decodes URL. a. The URL contains the Charging Station identity and optional EVSE number. b. (The URL format is known by CSMS and does not need to be configured in the Charging Station.) c. If URL validation or decoding fails no charging session will be started. *Use case ends here.* 4. *The remaining steps 4 to 9 are identical to scenario description #1 above.*

No.	Type	Description
	Scenario description #3	*Settlement when transaction has finished* 1. If cost calculated locally a. Charging Station sends a TransactionEventRequest with *eventType* = Ended and *costDetails* to CSMS. b. CSMS responds with TransactionEventResponse. c. Charging Station shows the cost breakdown in *costDetails* to EV Driver. 2. If cost calculated centrally a. Charging Station sends a TransactionEventRequest with *eventType* = Ended to CSMS. b. CSMS responds with TransactionEventResponse with *totalCost* and an *updatedPersonalMessage*. c. Charging Station shows the cost breakdown in *updatedPersonalMessage* to EV Driver. 3. CSMS performs settlement directly with PSP based on the <pspRef>. 4. PSP returns a receipt data or a URL to a receipt to CSMS. 5. CSMS may use SetDisplayMessageRequest to display a URL to the receipt to the EV Driver.
5	**Prerequisites**	Dynamic QR code: The Charging Station has been configured with dynamic QR code configuration variables in WebPaymentsCtrlr, and Charging Station periodically displays a QR code with a new time-based one-time password.
6	**Postcondition(s)**	CSMS remotely starts a transaction for EV Driver.
7	**Error handling**	If the time-based one-time password verfication fails, then CSMS will not start a transaction.
8	**Remark(s)**	The communication between CSMS and PSP, as well as the manner how a receipt is communicated to the EV Driver, is out of scope of this document. This use case resembles use case C24 "Ad hoc payment via stand-alone payment terminal" with the difference that the credit/debit card terminal is replaced by the web page that the URL in the QR code points to. **NOTE** The mechanism to provide a URL for web payment via a QR code, as described in this use case, also applies to other mechanisms that can provide a URL, such as NFC tags or Bluetooth Beacons. This use case assumes that the CSMS of the CSO handles URL validation and communication with PSP, as is also the case in the other ad hoc payment use cases (directly or indirectly via a payment terminal). It is also possible to involve an EMSP in the use case to handle payment. In that case the URL points to the EMSP system that executes steps 2 to 7 of the use case after which it requests CSMS (via a roaming protocol) to start a session on the charging station.

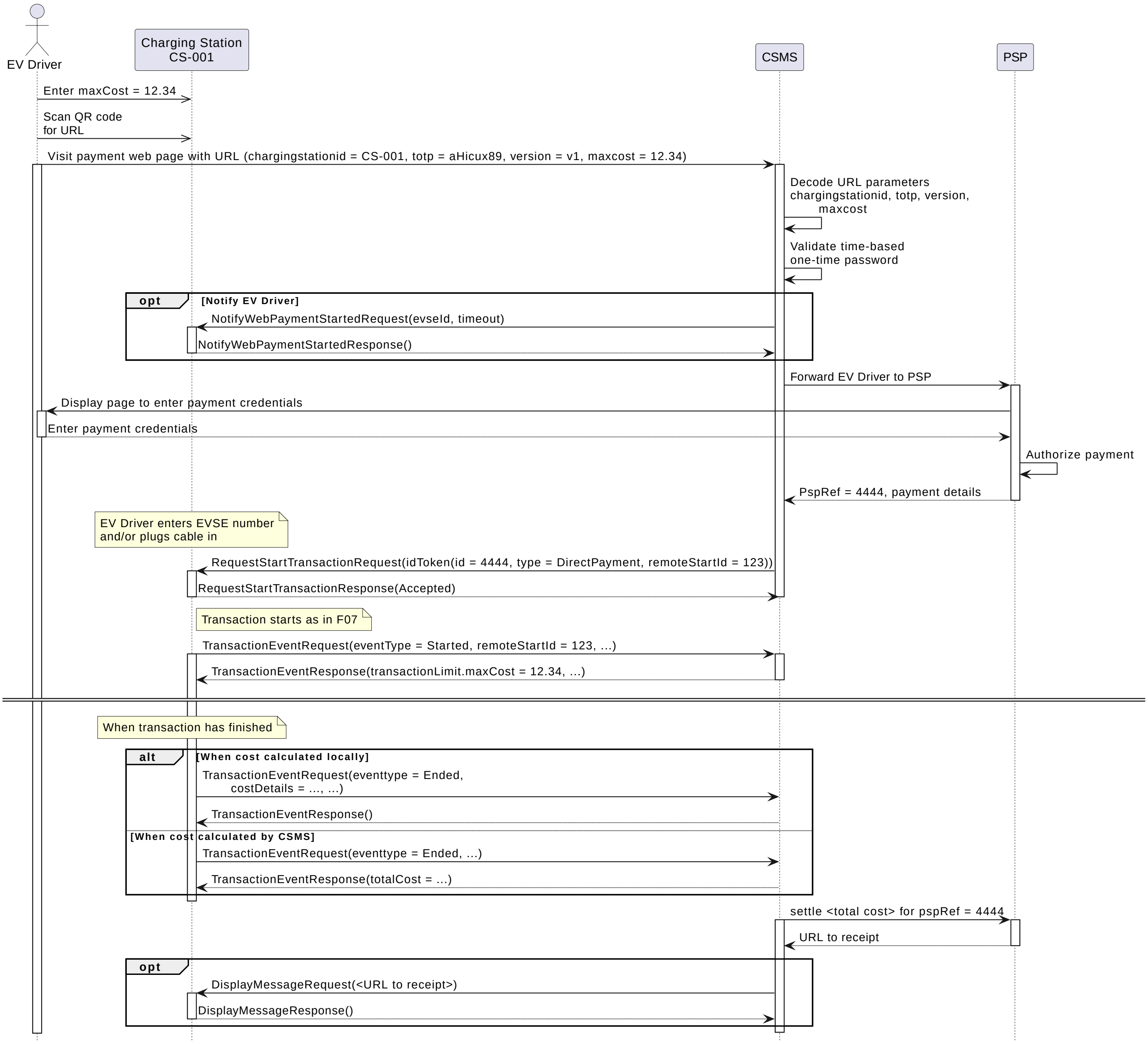

EV Driver
Charging Station
CS-001
CSMS
PSP
Enter maxCost = 12.34
Scan QR code
for URL
Visit payment web page with URL (chargingstationid = CS-001, totp = aHicux89, version = v1, maxcost = 12.34)
Decode URL parameters
chargingstationid, totp, version,
maxcost
Validate time-based
one-time password
opt
[Notify EV Driver]
NotifyWebPaymentStartedRequest(evseId, timeout)
NotifyWebPaymentStartedResponse()
Forward EV Driver to PSP
Display page to enter payment credentials
Enter payment credentials
Authorize payment
PspRef = 4444, payment details
EV Driver enters EVSE number
and/or plugs cable in
RequestStartTransactionRequest(idToken(id = 4444, type = DirectPayment, remoteStartId = 123))
RequestStartTransactionResponse(Accepted)
Transaction starts as in F07
TransactionEventRequest(eventType = Started, remoteStartId = 123, ...)
TransactionEventResponse(transactionLimit.maxCost = 12.34, ...)
When transaction has finished
alt
[When cost calculated locally]
TransactionEventRequest(eventtype = Ended,
costDetails = ..., ...)
TransactionEventResponse()
[When cost calculated by CSMS]
TransactionEventRequest(eventtype = Ended, ...)
TransactionEventResponse(totalCost = ...)
settle <total cost> for pspRef = 4444
URL to receipt
opt
DisplayMessageRequest(<URL to receipt>)
DisplayMessageResponse()

Figure 46. Ad hoc payment via QR code

URL definition for dynamic QR code

The QR code contains the URL to the ad hoc payment web page for the CSMS. This web page can be provided by a third party, like a PSP, but it is responsibility of the CSO. The format of the URL is up to the CSO, as long as certain mandatory URL parameters exist that are needed to support the use case.

The URL is a defined as a template that contains parts ("variables") that are filled in by the Charging Station. These variables are put between curly braces as path parameters in the URL template. For example:

Example URL without EVSE ID

```
https://qr.mycsms.com/{totp}/{version}/{chargingstationid}/
```

Example URL with EVSE

```
https://qr.mycsms.com/{chargingstationid}/{evse}/{totp}/{version}
```

The URL template is defined in WebPaymentsCtrlr.URLTemplate.

URL pointing to CSMS

In the situation as described in the use case the URL will point to the CSMS. CSMS will use a PSP to handle the payment, but CSMS is responsible for coordination and initiating the transaction. In the URL template the Charging Station will replace the variables:

- *chargingstationid* with its Charging Station identity,
- *evse* with the evse for which the payment is requested.
- *totp* with the calculated time-based one-time password,
- *version* with the version of the time-based one-time password algorithm,

A URL may contain additional query parameters that are filled in by the Charging Station based upon user input. This would allow an EV driver to specify, for example, a maximum amount of energy to charge. The following variables are defined, and when supported by the Charging Station, are reported in the variable named WebPaymentsCtrlr.URLParameters.

- *maxenergy* for maximum energy (in Wh) to be charged
- *maxtime* for maximum charging time (in seconds)
- *maxcost* for the maximum cost (in currency of charging station)

Example URL for EVSE #1 of CS1000 with maximum energy of 20 kWh

```
https://qr.mycsms.com/CS1000/1/aHicux89/v1/?maxenergy=20000
```

The only purpose of this is to communicate a maximum energy/time/cost request to CSMS before the authorization takes place. Once the transaction is started, this data will be communicated to CSMS as part of the *transactionLimit* field in the TransactionEventRequest.

URL pointing to external party

In the case that the URL points to an external party, such as an EMSP, the external party is responsible for coordination and initiating the start of a transaction. The external party will decode the URL.

Instead of a Charging Station identity, this URL will have to contain the roaming EVSE ID for the EVSE of the Charging Station. CSMS has to set the variable RoamingEvseId in WebPaymentsCtrlr to the roaming EVSE ID.

In the URL template the Charging Station will replace the variables:

- *roamingevseid* with the roaming EVSE id,
- *totp* with the calculated time-based one-time password,
- *version* with the version of the time-based one-time password algorithm,

Example URL template using a roaming ID of charging station

```
https://qr.mycsms.com/{roamingevseid}/{totp}/{version}
```

This requires a WebPaymentsCtrlr at the EVSE tier of the device model, i.e. each EVSE has its own WebPaymentsCtrlr such that CSMS can configure the roaming EVSE ID in WebPaymentCtrlr.RoamingEvseId for each EVSE.

The roaming EVSE ID needs to have been agreed upon by and communicated to the external party. A common standard is the format of eMI3/ISO15118 [ref-ISOIEC15118-2].

C25 - Ad hoc payment via static or dynamic QR code - Requirements

NOTE For a static QR code any requirements related to a one-time password (TOTP) are not applicable.

ID	Precondition	Requirements	Note
URL validation for dynamic QR code			
C25.FR.01	When component WebPaymentsCtrlr exists and has "Enabled" = true	Charging Station SHALL display a dynamic QR code according to configuration variables in WebPaymentsCtrlr.	See URL definition for dynamic QR code for description of URL template.

ID	**Precondition**	**Requirements**	**Note**
C25.FR.03	If Charging Station supports WebPaymentsCtrlr.URLParameters	Charging Station SHALL allow input from EV Driver for these parameters (e.g. maximum time, energy or cost) to be added as query parameters to the URL in the QR code.	This requires that the users inputs values before the QR code is displayed.
C25.FR.04	C25.FR.03 AND URL parameter "maxtime" is supported EV Driver specified a maximum time	CS SHALL append the query variable "maxtime" with the time in seconds to the URL and enforce the maximum transaction duration.	This also means that CS will add a *transactionLimit.maxTime* for this value to the TransactionEventRequest messages. (See E16).
C25.FR.05	C25.FR.03 AND URL parameter "maxenergy" is supported EV Driver specified a maximum energy to charge	CS SHALL append the query variable "maxenergy" with the energy in Wh and enforce the maximum charging energy.	This also means that CS will add a *transactionLimit.maxEnergy* for this value to the TransactionEventRequest messages. (See E16).
C25.FR.06	C25.FR.03 AND URL parameter "maxcost" is supported EV Driver specified a maximum cost for charging	CS SHALL append the query variable "maxcost" to the URL and enforce the maximum charging cost.	This also means that CS will add a *transactionLimit.maxCost* for this value to the TransactionEventRequest messages. (See E16).
C25.FR.07	When EV Driver visits the web page of CSMS using the URL in the QR code	CSMS SHALL validate the time-based one-time password in the URL	See description in Validation of TOTP.
C25.FR.08	If the time-based one-time password in the URL is not valid	CSMS SHALL NOT forward EV Driver to a web page of PSP to enter payment credentials and SHALL NOT authorize EV Driver to charge	No payment is authorized and no transaction is started.
C25.FR.09	If the time-based one-time password in the URL is valid	CSMS SHALL decode the URL parameters as defined in URL definition for dynamic QR code.	URL parameters must at least include an identity of the Charging Station and optionally an EVSE number.
Payment authorization			
C25.FR.20	If URL successfully decoded	CSMS SHALL forward EV Driver to a web page of PSP to enter payment credentials	The protocol between PSP and CSMS is not specified in this document.
C25.FR.21	C25.FR.20	CSMS MAY send a NotifyWebPaymentStartedRequest message with *evseId* having the EVSE number that was decoded from the URL and *timeout* the time in seconds that Charging Station may lock the EVSE to prevent locally started transactions.	After *timeout* seconds Charging Station should free the EVSE again for local transactions.
C25.FR.22	C25.FR.21	Charging Station SHOULD provide feedback to EV Driver that the QR code was successfully scanned and SHOULD prevent further local interactions with the EVSE referred to in *evseId* for as long as *timeout* seconds or until a RequestStartTransactionRequest is received from CSMS.	
C25.FR.23	When CSMS receives a message from Payment Kiosk that payment is authorized with authorization reference <PspRef> for Charging Station with identity <CS> and optionally an EVSE <evse>	CSMS SHALL send a RequestStartTransactionRequest to Charging Station with identity <CS> and EVSE <evse> (when present) and having *idToken.idToken* = <PspRef> and *idToken.type* = `DirectPayment`	There is need to include *additionalInfo* elements for payment details, like cardBin, cardLast4Digits, towards Charging Station. The protocol between Payment Kiosk and CSMS is not specified in this document.

ID	Precondition	Requirements	Note
C25.FR.24	C25.FR.23 AND Charging Station sends a TransactionEventRequest with *eventType* = `Started`	CSMS SHALL respond with a TransactionEventResponse with *transactionLimit* set to the required limit.	*transactionLimit* uses *maxCost* for cost limit, *maxEnergy* for energy limit, *maxTime* for time limit or *maxSoC* for SoC limit.
C25.FR.25	C25.FR.24	Charging Station SHALL include this limit in the field *transactionInfo.transactionLimit* (once) in the next TransactionEventRequest	
C25.FR.26		Charging Station SHALL behave as in use case E16 - Transactions with fixed cost, energy, SoC or time.	
C25.FR.27	Upon receiving a NotifyWebPaymentStartedRequest	Charging Station SHALL respond with a NotifyWebPaymentStartedResponse without any parameters.	
Settlement			
C25.FR.40	After CSMS received a TransactionEventRequest with *eventType* = Ended or the transaction	CSMS SHALL send a message to PSP to settle the cost	Cost may have been calculated locally or centrally.
C25.FR.41	When CSMS receives receipt information or a URL to a receipt from PSP	CSMS MAY send a SetDisplayMessageRequest with a URL where EV Driver can download the receipt .	Other methods to find and download receipts may also be offered by CSO.
URL requirements for dynamic QR code			
C25.FR.50		The URL template for the dynamic QR code SHALL contain the following placeholder for URL parameters: • **{chargingstationid}** or **{roamingevseid}** • **{totp}** for the generated TOTP • **{version}** for the TOTPVersion	
C25.FR.51	If the WebPaymentsCtrlr.URLTemplate contains **{chargingstationid}**	Charging Station SHALL replace placeholder **{chargingstationid}** in the URL template by the value of SecurityCtrlr.Identity	
C25.FR.52	If the WebPaymentsCtrlr.URLTemplate contains **{evse}**	Charging Station SHALL replace placeholder **{evse}** in the URL template by the number of the EVSE for which the payment request is done	
C25.FR.53	If the WebPaymentsCtrlr.URLTemplate contains **{roamingevseid}**	Charging Station SHALL replace placeholder **{roamingevseid}** in the URL template by the value of WebPaymentsCtrlr.RoamingEvseId	
C25.FR.56	If Charging Station wishes to communicate a maximum energy amount for the QR code authorization	Charging Station SHALL add the query parameter "**maxenergy**=XXX" to the URL, where XXX is the maximum energy (in Wh) to be charged.	
C25.FR.57	If Charging Station wishes to communicate a maximum charging time for the QR code authorization	Charging Station SHALL add the query parameter "**maxtime**=XXX" to the URL, where XXX is the maximum charging time (in seconds).	
C25.FR.58	If Charging Station wishes to communicate a maximum cost for the QR code authorization	Charging Station SHALL add the query parameter "**maxcost**=XXX" to the URL, where XXX is the maximum cost (in currency of tariff at charging station).	
C25.FR.59		CSMS SHALL accept the query parameters maxenergy, maxtime and maxcost in a QR code authorization URL.	
C25.FR.60		CSMS MAY ignore a non-supported query parameter in the QR code authorization URL.	If a maximum limit was provided, Charging Station will apply this to the transaction, regardless whether CSMS has support for it.

Time-based One-Time password algorithm

The algorithm to calculate the time-based one-time password uses the HMAC-SHA256 hash function on the byte representation of the start of the current time interval in seconds. The bytes are mapped to digits and characters. For added security the mapping starts not at the first byte, but at a semi-random offset that is calculated from the last byte of the hash.

The configuration parameters for the time-based one-time password are set in the WebPaymentsCtrlr: ValidityTime, SharedSecret and Length.

TOTP algorithm, version 1

The algorithm is defined in pseudocode as follows:

TOTP v1 algorithm in pseudocode

```
// Input parameters
set validityTime to <parameter ValidityTime>
set totpLength to <parameter Length>
set sharedSecret to <parameter SharedSecret>

set base62 to "0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ"

// Get time interval
calculate timeInSeconds as current UTC time - Unix Epoch (01-01-1970)
calculate timeInterval as floor of timeInSeconds / validityTime
convert timeInterval to bytes and store in timeBytes

if system's byte order is Little Endian then
    reverse byte order of timeBytes

// Calculate hash
create new HMAC-SHA256 hash function with sharedSecret as the key (converted to bytes using UTF-8 encoding)
calculate hash of timeBytes using HMAC-SHA256 and store in hash variable

// Convert hash bytes to characters starting at random offset
calculate offset as (last byte of hash) bitwise AND 0x0F
create a new empty string builder
for i from 0 to totpLength do
    find the character at position (hash value at index (offset + i) modulo length of hash)
        modulo length of base62 in the base62 string
    append this character to the string builder

end for

return string builder as string
```

The following is a C# version of this algorithm:

TOTP v1 algorithm in C#

```
// Input parameters
var validityTime    = TimeSpan.FromSeconds(ValidityTime); // parameter from URL
var totpLength      = Length;       // parameter from URL
var sharedSecret    = SharedSecret; // parameter from URL
var base62          =
"0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ";

// Get time interval
var timeInSeconds   = DateTime.UtcNow - new DateTime(1970, 1, 1);
var timeInterval    = (UInt64) (timeInSeconds.TotalSeconds /
validityTime.TotalSeconds);
var timeBytes       = BitConverter.GetBytes(timeInterval);

if (BitConverter.IsLittleEndian)
    Array.Reverse(timeBytes);

using (var hmac = new HMACSHA256(Encoding.UTF8.GetBytes(sharedSecret)))
{
    // Calculate hash
    var hash = hmac.ComputeHash(timeBytes);

    // Convert hash bytes to characters starting at random offset
    var offset         = hash[^1] & 0x0F;
    var stringBuilder  = new StringBuilder();
    for (var i = 0; i < totpLength; i++)
        stringBuilder.Append(base62[hash[(offset + i) % hash.Length] %
base62.Length]);

    return stringBuilder.ToString();
}
```

Validation of TOTP

CSMS validates the time-based one-time password (TOTP) in the URL against the TOTP that CSMS calculates for the current time.

The TOTP in the URL (URL-TOTP) is considered valid if it equals the TOTP calculated by CSMS (CSMS-TOTP). If the TOTPs are not equal CSMS shall try to compare the URL-TOTP against a CSMS-TOTP for the previous time interval and the next time interval. This overcomes failed validation as a result processing delays or small differences in clock time.

If URL-TOTP and CSMS-TOTP still do not match in any of these cases, then CSMS shall consider the URL as invalid.

D. Local Authorization List Management

Chapter 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.

Chapter 2. Use cases & Requirements

D01 - Send Local Authorization List

No.	Type	Description
1	**Name**	Send Local Authorization List
2	**ID**	D01
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	**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)**	**Successful postcondition:** - A new Local Authorization List is installed on the Charging Station. **Failure postcondition:** - The Local Authorization List on the Charging Station stays as it was. - If the status is *Failed* or *VersionMismatch*.
7	**Error handling**	If the status is *Failed* or *VersionMismatch* and the updateType 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 updateType *Full* and adding identifiers using updateType *Differential* until the list is completely sent (the amount of identifiers that can be sent in a single SendLocalListRequest is limited as described in requirement D01.FR.11).
8	**Remark(s)**	n/a

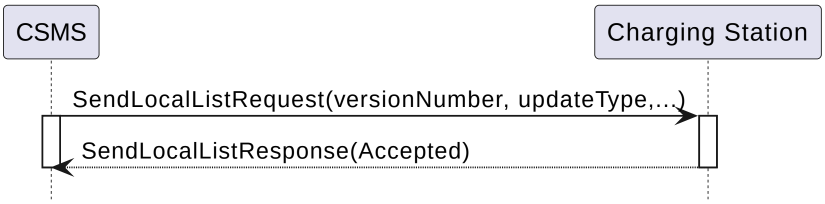

CSMS
Charging Station
SendLocalListRequest(versionNumber, updateType,...)
SendLocalListResponse(Accepted)

Figure 47. Sequence Diagram: Send Local Authorization List

D01 - Send Local Authorization List - Requirements

Table 60. D01 - Requirements

ID	Precondition	Requirement definition	Note
D01.FR.01		SendLocalListRequest SHALL contain the type of update (*updateType*) and a version number (*versionNumber*) that the Charging Station MUST associate with the Local Authorization List after it has been updated.	
D01.FR.02		SendLocalListResponse 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* 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 *updateType* is `Differential` AND *localAuthorizationList* contains AuthorizationData elements without idTokenInfo	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		*versionNumber* in a SendLocalListRequest SHALL be greater than 0.	In GetLocalListVersionResponse the *versionNumber* = 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 *updateType* = `Differential` AND *versionNumber* 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 *status* set to `VersionMismatch`.	

D02 - Get Local List Version

No.	Type	Description
1	**Name**	Get Local List Version
2	**ID**	D02
	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.
7	**Error handling**	n/a
8	**Remark(s)**	A *versionNumber* 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. In contrast, a local authorization list that was emptied, because CSMS sent a SendLocalListRequest with an empty *localAuthorizationList*, does have a *versionNumber* > 0.

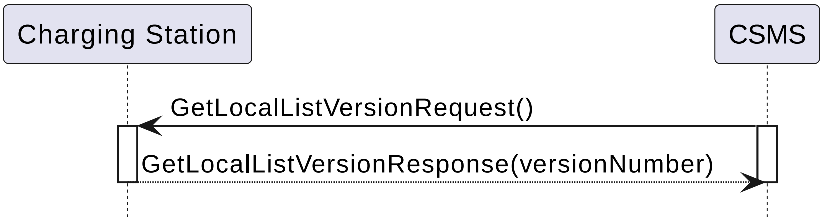

Charging Station
CSMS
GetLocalListVersionRequest()
GetLocalListVersionResponse(versionNumber)

Figure 48. Sequence Diagram: Get Local List Version

D02 - Get Local List Version - Requirements

Table 61. 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 *versionNumber* 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 *versionNumber* is 0 (zero) to indicate that there is no Local Authorization List.

E. Transactions

Chapter 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 2.1 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)

For the 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 2.1 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.

Table 62. The settings for an OCPP 1.6 compatible transaction

Configuration Variable	Values
`TxStartPoint`	PowerPathClosed
`TxStopPoint`	EVConnected, Authorized

1.2. TransactionId generation

New in OCPP 2.x generation: 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 TxUpdatedInterval == 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*. This implies that a TransactionEventRequest(*eventType* = `Started`) always has a *chargingState*, because the state goes from non-existent to a value.
> If the change of charging state is the only event, then TransactionEventRequest has a *triggerReason* = `ChargingStateChanged`, but if a change in charging state occurs together with other events, such as those represented by *triggerReason* `CablePluggedIn` or `(Stop)Authorized`, for example, then *chargingState* can simply be reported as part of that message.
> A TransactionEventRequest with *triggerReason* = `ChargingStateChanged` must contain *chargingState*.

transactionInfo.stoppedReason

> (B12.FR.04, B12.FR.08) *stoppedReason* `ImmediateReset` is sent when transaction ended as result of a reset.
> (C15.FR.04, C16.FR.08) *stoppedReason* `DeAuthorized` or `MasterPass`
> (E03.FR.05, E05.FR.10, E06.FR.08/09, E07.FR.04/05/06, E15.FR.04, E17.FR.21/22, E18.FR.25) The *stoppedReason* must be provided in the TransactionEventRequest(*eventType*=`Ended`), unless the value is `Local`, in which case it may be omitted.
> (F02.FR.07, F03.FR.03, 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`.
(K18.FR.23, K19.FR.16, Q01.FR.05) *stoppedReason* `ReqEnergyTransferRejected` is sent when transaction is ended, because requested energy transfer type is not granted.

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*.

transactionInfo.operationMode

(K01.FR.110) Whenever the operation mode changes, the Charging Station must send a TransactionEventRequest containing an *transactionInfo* element with *operationMode*. A TransactionEventRequest with *triggerReason* = `OperationModeChanged` must contain *operationMode*.

transactionInfo.transactionLimits

(E16.FR.02, E16.FR.03, E16.FR.10, C17.FR.03, C18.FR.06) When CSMS or Charging Station provide a limit in time, energy, SoC or cost, then this must be included once in the *transactionInfo* element.

transactionInfo.tariffId

(I08.FR.13) When a tariff applies to the transaction, then the id of the tariff in use, is added to the *transactionInfo* element.

costDetails

(I12.FR.01, I12.FR.02) When Charging Station has calculated transaction cost locally, it must provide the field *costDetails* in TransactionEventRequests.

Chapter 2. Use cases & Requirements

2.1. OCPP transaction mechanism

E01 - Start Transaction options

No.	Type	Description
1	**Name**	Start Transaction options
2	**ID**	E01
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 eventType = Started), 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	**1.** The EV Driver parks his "EV" at a Charging Station with a parking bay occupancy detector, which triggers the detector. **2.** The Charging Station sends a TransactionEventRequest (eventType = Started) notifying the CSMS about a transaction that has started (even when the driver is not yet known. **3.** The CSMS responds with a TransactionEventResponse, confirming that the TransactionEventRequest was received.
	Prerequisite(s)	No transaction is ongoing on the EVSE. Configuration Variable: `TxStartPoint` contains: ParkingBayOccupancy
	Postcondition(s)	**Successful postcondition:** The transaction is ongoing and the CSMS is *Successfully* informed. **Failure postcondition:** The transaction is *not* ongoing, *or* The CSMS is *not* informed.

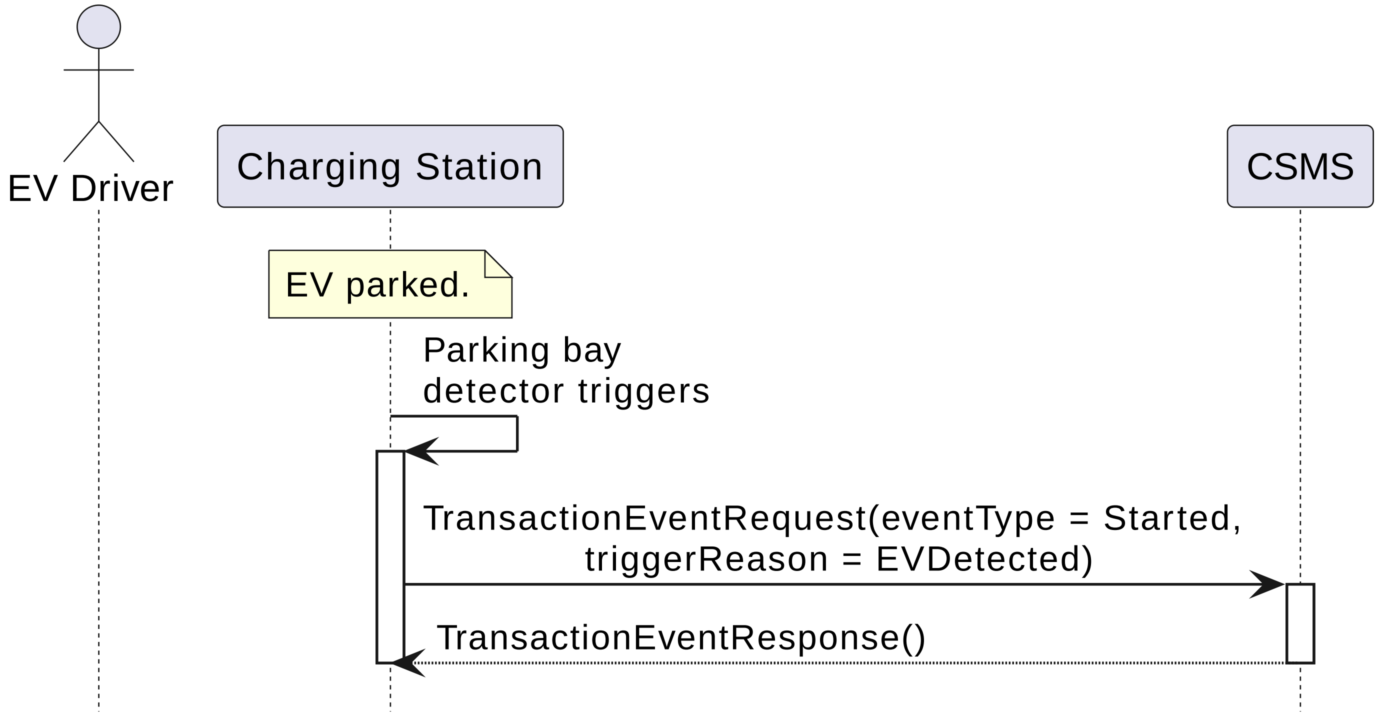

EV Driver
Charging Station
CSMS
EV parked.
Parking bay detector triggers
TransactionEventRequest(eventType = Started, triggerReason = EVDetected)
TransactionEventResponse()

Figure 49. 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	**1.** The Charging Station sets up a connection with the EV. **2.** The Charging Station sends a TransactionEventRequest (eventType = Started) notifying the CSMS about a transaction that has started (even when the driver is not yet known). **3.** The CSMS responds with a TransactionEventResponse, confirming that the TransactionEventRequest was received.
	Prerequisite(s)	No transaction is ongoing on the EVSE. Configuration Variable: `TxStartPoint`: EVConnected
	Postcondition(s)	**Successful postcondition:** The transaction is ongoing and the CSMS is *Successfully* informed. **Failure postcondition:** The transaction is *not* ongoing, *or* The CSMS is *not* informed.

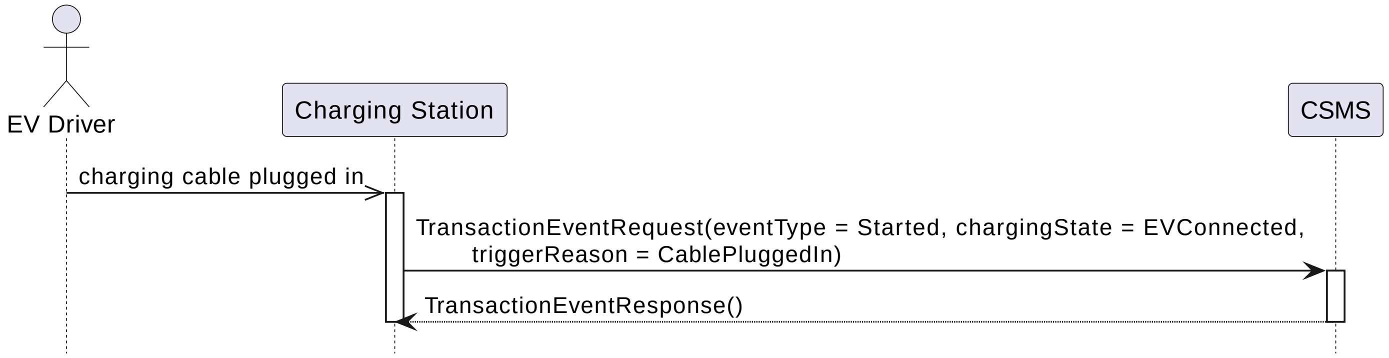

EV Driver
Charging Station
CSMS
charging cable plugged in
TransactionEventRequest(eventType = Started, chargingState = EVConnected, triggerReason = CablePluggedIn)
TransactionEventResponse()

Figure 50. Sequence Diagram: Start Transaction options - EVConnected

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

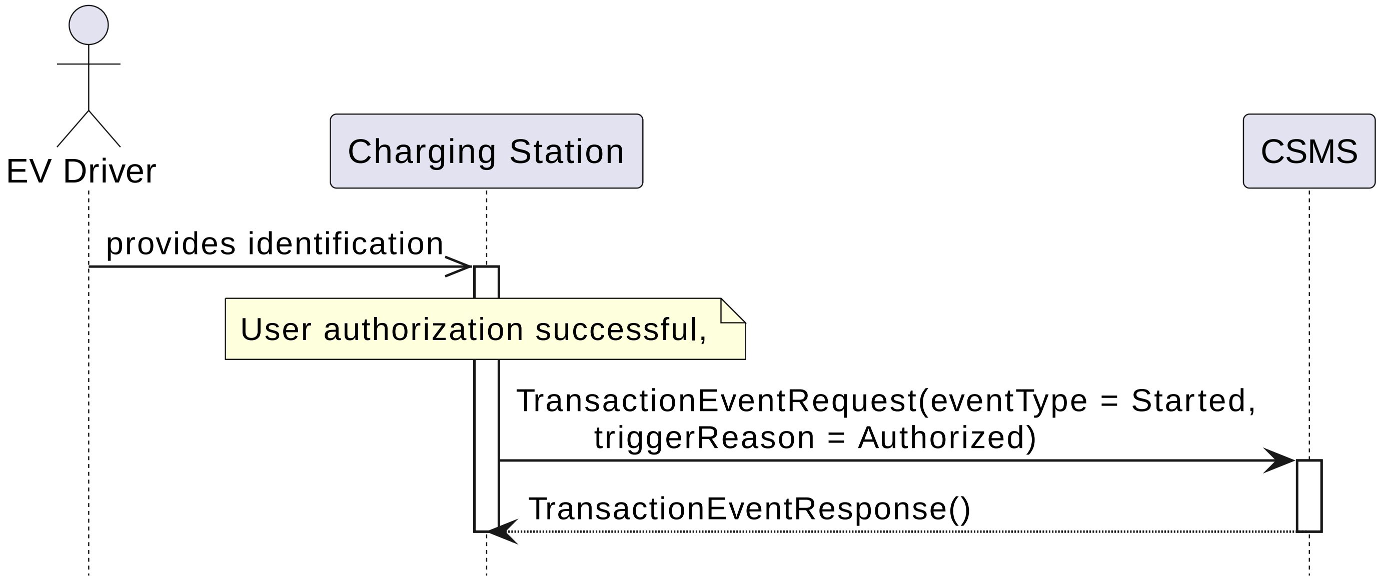

EV Driver
Charging Station
CSMS
provides identification
User authorization successful,
TransactionEventRequest(eventType = Started, triggerReason = Authorized)
TransactionEventResponse()

Figure 51. 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	**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 (eventType = Started) notifying the CSMS about a transaction that has started. **5.** The CSMS responds with a TransactionEventResponse, confirming that the TransactionEventRequest was received.
	Prerequisite(s)	No transaction is ongoing on the EVSE. Configuration Variable: `TxStartPoint`: DataSigned. The Charging Station has a meter that can sign measured values Configuration Variable: `SampledDataSignReadings` set to *true*.
	Postcondition(s)	**Successful postcondition:** The transaction is ongoing and the CSMS is *Successfully* informed. **Failure postcondition:** The transaction is *not* ongoing, *or* The CSMS is *not* informed.

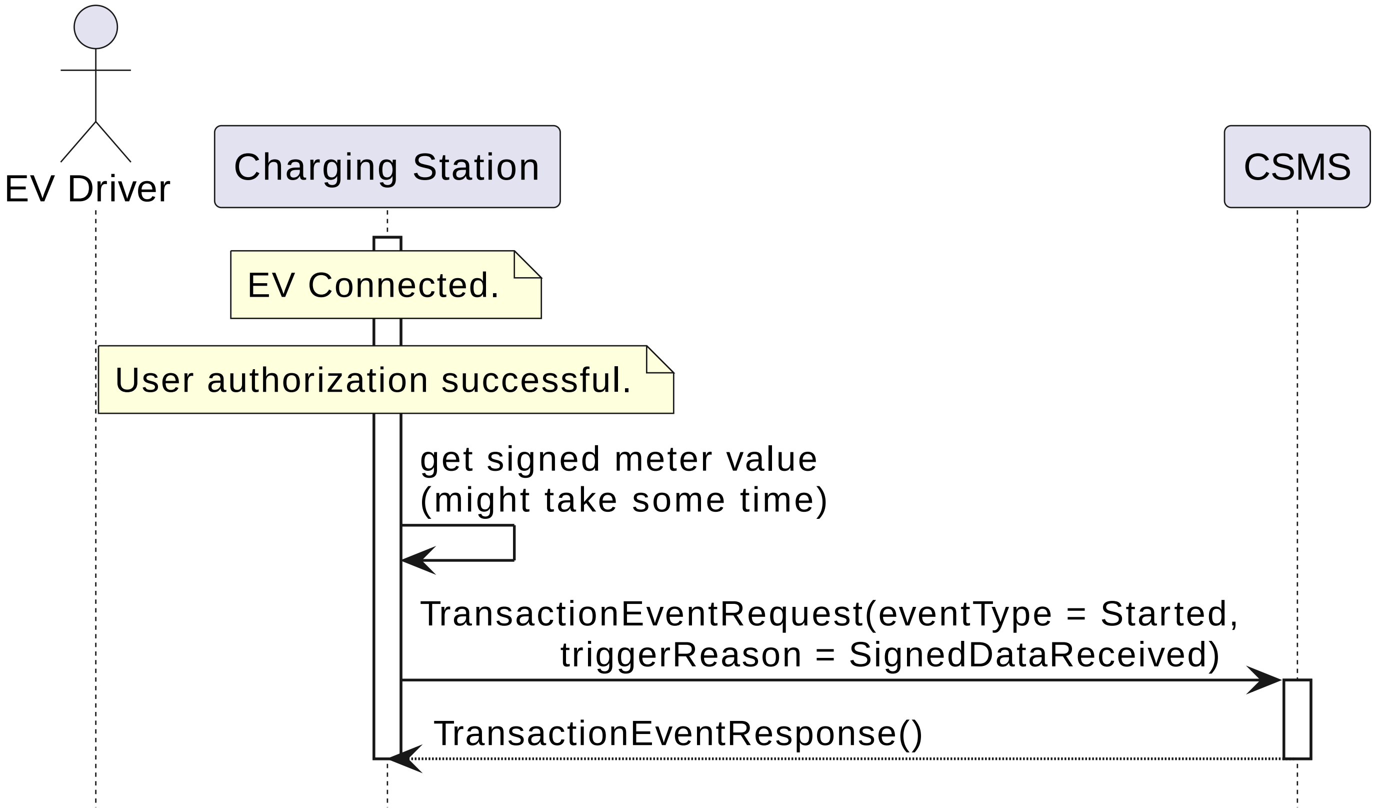

EV Driver
Charging Station
CSMS
EV Connected.
User authorization successful.
get signed meter value
(might take some time)
TransactionEventRequest(eventType = Started,
triggerReason = SignedDataReceived)
TransactionEventResponse()

Figure 52. 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 transfered.**
	Scenario description	**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 TransactionEventRequest (eventType = Started) notifying the CSMS about a transaction that has started. **4.** The CSMS responds with a TransactionEventResponse, confirming that the TransactionEventRequest was received.
	Prerequisite(s)	No transaction is ongoing on the EVSE. Configuration Variable: `TxStartPoint`: PowerPathClosed. Charging Cable plugged in.
	Postcondition(s)	**Successful postcondition:** The transaction is ongoing and the CSMS is *Successfully* informed. **Failure postcondition:** The transaction is *not* ongoing, *or* The CSMS is *not* informed.

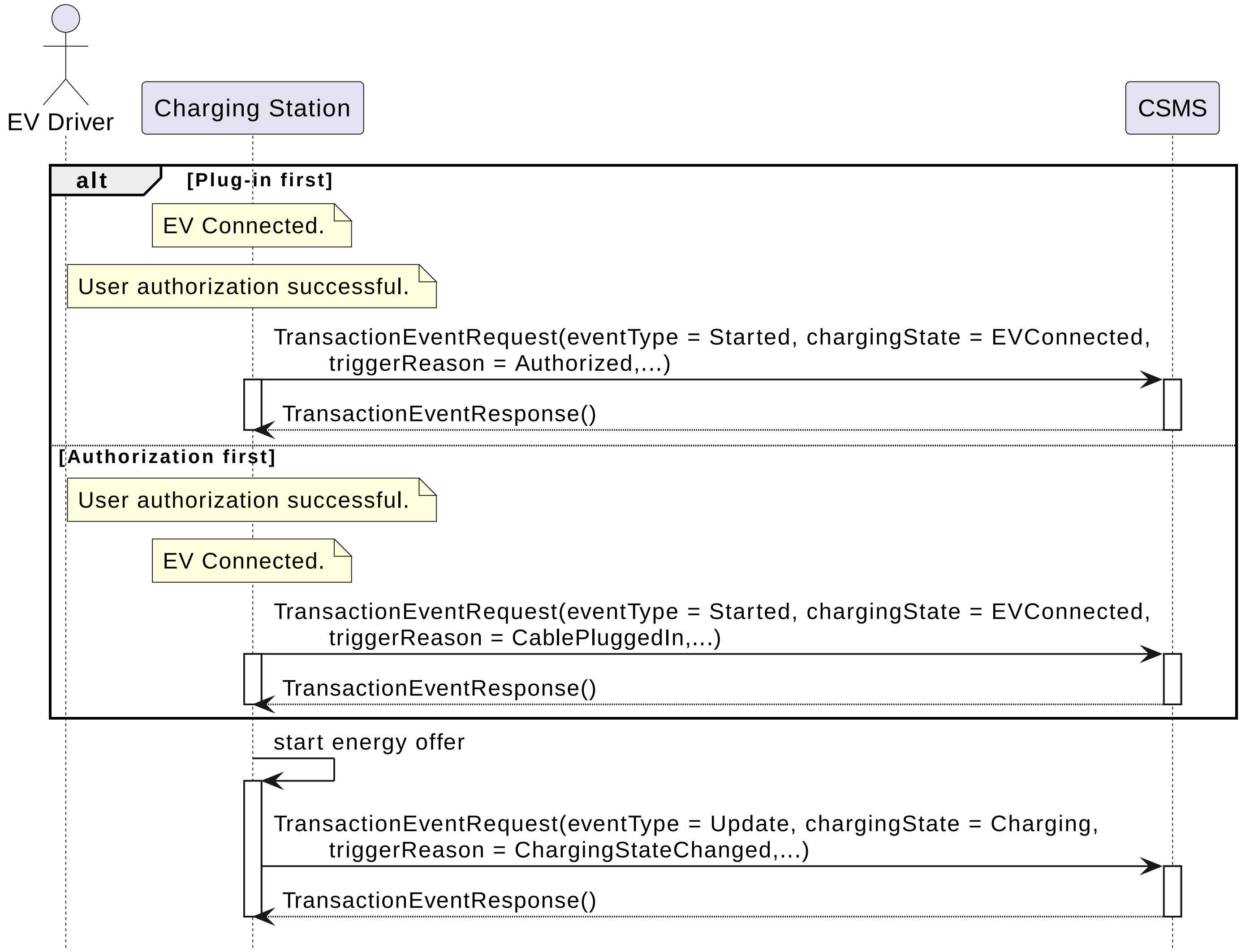

EV Driver
Charging Station
CSMS
alt
[Plug-in first]
EV Connected.
User authorization successful.
TransactionEventRequest(eventType = Started, chargingState = EVConnected, triggerReason = Authorized,...)
TransactionEventResponse()
[Authorization first]
User authorization successful.
EV Connected.
TransactionEventRequest(eventType = Started, chargingState = EVConnected, triggerReason = CablePluggedIn,...)
TransactionEventResponse()
start energy offer
TransactionEventRequest(eventType = Update, chargingState = Charging, triggerReason = ChargingStateChanged,...)
TransactionEventResponse()

Figure 53. Sequence Diagram: Start Transaction options - PowerPathClosed

S6	*Scenario objective*	**To start a transaction when the energy flow starts.**
	Scenario description	**1.** The EV Driver is authorized by the Charging Station and/or CSMS. **2.** The Charging Station closes the power relay. **3.** The EV starts charging, energy flow starts. **4.** The Charging Station sends a TransactionEventRequest (eventType = Started) notifying the CSMS about a transaction that has started. **5.** The CSMS responds with a TransactionEventResponse, confirming that the TransactionEventRequest was received.
	Prerequisite(s)	Configuration Variable: `TxStartPoint`: EnergyTransfer.
	Postcondition(s)	**Successful postcondition:** The transaction is ongoing and the CSMS is *Successfully* informed. **Failure postcondition:** The transaction is *not* ongoing, *or* The CSMS is *not* informed.

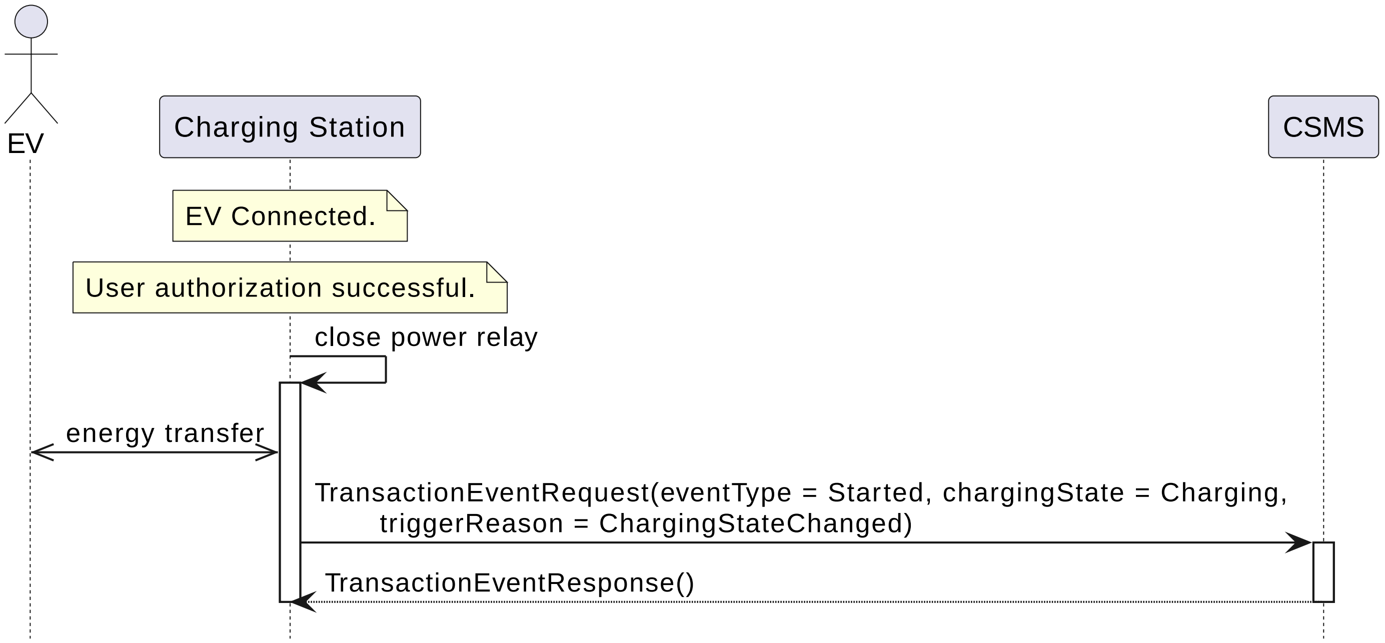

EV
Charging Station
CSMS
EV Connected.
User authorization successful.
close power relay
energy transfer
TransactionEventRequest(eventType = Started, chargingState = Charging, triggerReason = ChargingStateChanged)
TransactionEventResponse()

Figure 54. Sequence Diagram: Start Transaction options - EnergyTransfer

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

E01 - Start Transaction options - Requirements

Table 63. E01 - Requirements

ID	Precondition	Requirement definition
E01.FR.01	`TxStartPoint` contains: ParkingBayOccupancy AND Parking Bay Detector detects an "EV" AND No transaction has started yet	The Charging Station SHALL start a transaction and send a TransactionEventRequest (eventType = Started) to the CSMS.
E01.FR.02	`TxStartPoint` contains: EVConnected 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 TransactionEventRequest (eventType = Started) to the CSMS.
E01.FR.03	`TxStartPoint` contains: Authorized AND The EV Driver is authorized AND No transaction has started yet	The Charging Station SHALL start a transaction and send a TransactionEventRequest (eventType = Started) to the CSMS.

ID	Precondition	Requirement definition
E01.FR.04	`TxStartPoint` contains: DataSigned AND The Charging Station has a meter that can sign measured values AND Configuration Variable: `SampledDataSignReadings` set to *true*. 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 TransactionEventRequest (eventType = Started) to the CSMS.
E01.FR.05	`TxStartPoint` contains: PowerPathClosed 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 TransactionEventRequest (eventType = Started) to the CSMS.
E01.FR.06	`TxStartPoint` contains: EnergyTransfer AND Energy flow starts AND No transaction has started yet on this EVSE	The Charging Station SHALL start a transaction and send a TransactionEventRequest (eventType = Started) to the CSMS.
E01.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.
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.
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*.

ID	Precondition	Requirement definition
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

No.	Type	Description
1	**Name**	Start Transaction - Cable Plugin First
2	**ID**	E02
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
	Scenario description	**1.** The EV Driver plugs in the cable at the Charging Station. **2.** The Charging Station sends a NotifyEventRequest with component.name *Connector*, variable.name *AvailabilityState* and actualValue *Occupied* to the CSMS to inform it about a Connector that became *Occupied* **3.** The Charging Station sends a TransactionEventRequest (eventType = Started) notifying the CSMS about a transaction that has started (even when the driver is not yet known.) **4.** The CSMS responds with a TransactionEventResponse, confirming that the TransactionEventRequest was received. **5.** The EV Driver is authorized by the Charging Station and/or CSMS. **6.** The energy offer starts. **7.** The Charging Station sends a TransactionEventRequest (eventType = Updated) with the authorized idToken information to the CSMS to inform about the charging status and which idToken belongs to the transaction. **8.** The CSMS responds with a TransactionEventResponse to the Charging Station with the IdTokenInfo.status *Accepted*. **9.** During the charging process, the Charging Stations continues to send TransactionEventRequest (Updated) messages for transaction-related notifications.
	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)**	**Successful postcondition:** The transaction is ongoing and the CSMS is *Successfully* informed. **Failure postcondition:** The transaction is *not* ongoing. *or* The CSMS is *not* informed. *or* Start Transaction - Id not accepted.
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.

No.	Type	Description
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 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 message are sent. For more details see the use cases: E01 - Start Transaction options and E06 - Stop Transaction options.

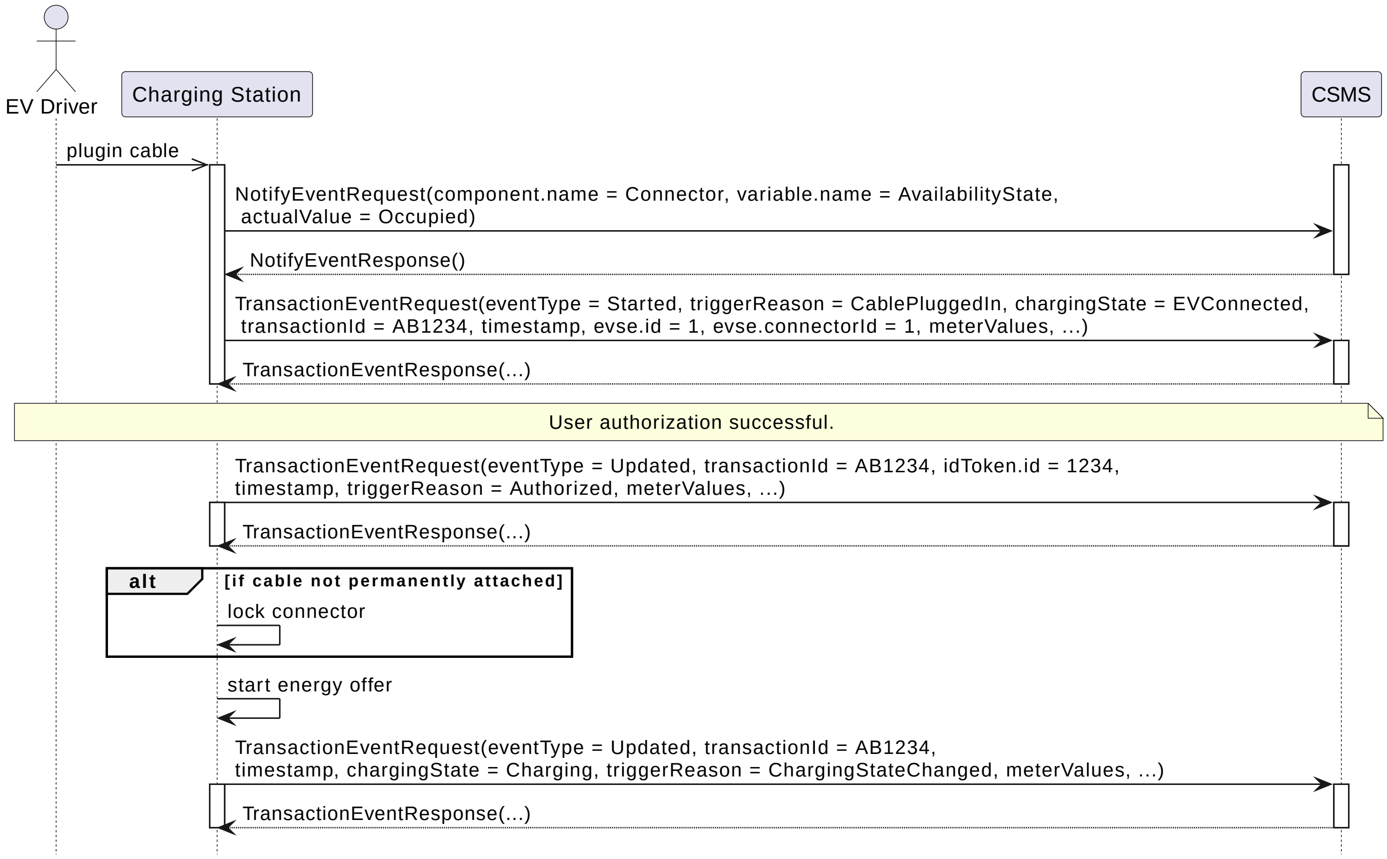

EV Driver
Charging Station
CSMS
plugin cable
NotifyEventRequest(component.name = Connector, variable.name = AvailabilityState, actualValue = Occupied)
NotifyEventResponse()
TransactionEventRequest(eventType = Started, triggerReason = CablePluggedIn, chargingState = EVConnected, transactionId = AB1234, timestamp, evse.id = 1, evse.connectorId = 1, meterValues, ...)
TransactionEventResponse(...)
User authorization successful.
TransactionEventRequest(eventType = Updated, transactionId = AB1234, idToken.id = 1234, timestamp, triggerReason = Authorized, meterValues, ...)
TransactionEventResponse(...)
alt
[if cable not permanently attached]
lock connector
start energy offer
TransactionEventRequest(eventType = Updated, transactionId = AB1234, timestamp, chargingState = Charging, triggerReason = ChargingStateChanged, meterValues, ...)
TransactionEventResponse(...)

Figure 55. Sequence Diagram: Start Transaction - Cable Plugin First

E02 - Start Transaction - Cable Plugin First - Requirements

Table 64. 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. (Same as E01.FR.13)
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.
E02.FR.05 **(2.1)**	When a cable is plugged in	The Charging Station SHALL send a NotifyEventRequest message for **component**(name = 'Connector', evse.id = *<x>*, evse.connectorId = *<y>*), **variable**(name = 'AvailabilityState'), and **actualValue** = 'Occupied' to signal that Connector *<y>* of EVSE *<x>* is now occupied.	Alternatively, the old StatusNotificationRequest with status: *Occupied* can still be sent.
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. (Same as E01.FR.07)

ID	Precondition	Requirement definition	Note
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.	(Same as E01.FR.08)
E02.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.	(Same as E01.FR.09)
E02.FR.10	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.	(Same as E01.FR.11)
E02.FR.11	E02.FR.10 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*.	(Same as E01.FR.14)
E02.FR.13	If the charging state changes	The Charging Station SHALL send a TransactionEventRequest including the chargingState element.	(Same as E01.FR.18)
E02.FR.14	`SampledDataSignReadings` is *true*	The Charging Station SHALL retrieve signed meter values and put them in the *signedMeterValue* field of sampledValues.	
E02.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.	(Same as E01.FR.15)
E02.FR.16	After a transaction has been started	The Charging Station MAY send additional TransactionEventRequest(eventType = Updated) messages during the transaction when a trigger event occurs.	
E02.FR.17	When a transaction-related trigger event occurs, listed in *TriggerReasonEnumType* 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 *CablePluggedIn* should be used, not *ChargingStateChanged*. It is not forbidden to send separate TransactionEventRequest messages for each trigger, though.
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 *numberOfPhasesUsed* 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 *numberOfPhasesUsed* field, in the next TransactionEventRequest.	

ID	Precondition	Requirement definition	Note
E02.FR.20	When a transaction has not been authorized before AND the Charging Station authorizes an *idToken* to start charging	The next TransactionEventRequest from Charging Station SHALL contain the *idToken* and have *triggerReason* = `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 (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.	(Same as E01.FR.17)

E03 - Start Transaction - IdToken First

No.	Type	Description
1	**Name**	Start Transaction - IdToken First
2	**ID**	E03
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
	Scenario description	**1.** The EV Driver is authorized by the Charging Station and/or CSMS. **2.** The Charging Station informs the CSMS that a transaction has started by sending a TransactionEventRequest (eventType = Started). **3.** The EV Driver plugs in the Charging Cable at the Charging Station. **4.** The Charging Station sends NotifyEventRequest with component.name *Connector*, variable.name *AvailabilityState* and actualValue *Occupied* to, and receives NotifyEventResponse from the CSMS. **5.** The Charging Station informs the CSMS that the EV started charging by sending a TransactionEventRequest (eventType = Updated, chargingState = Charging). **6.** The CSMS responds with TransactionEventResponse, accepting the transaction.
5	**Prerequisite(s)**	IdToken is presented prior to plugin cable.
6	**Postcondition(s)**	**Successful postcondition:** A transaction is started and the ChargingState is *Charging* **Failure postcondition:** No transaction is started
7	**Error handling**	n/a
8	**Remark(s)**	It is likely that the CSMS applies sanity checks to the data contained in TransactionEventRequest messages it received. The outcome of such sanity checks SHALL 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. The scenario description and sequence diagram above are based on the Configuration Variable for start transaction being configured as follows: `TxStartPoint`: Authorized 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.

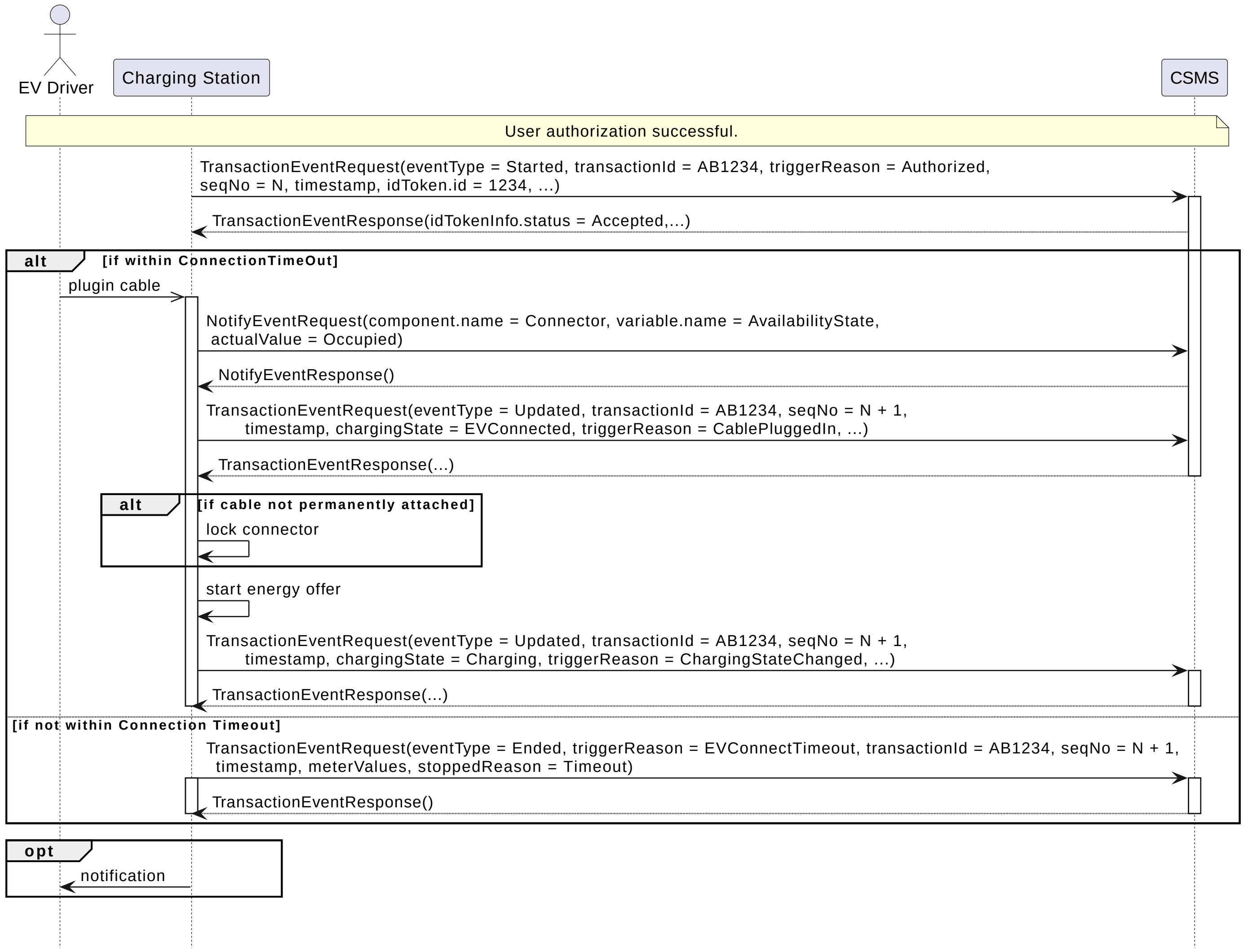

EV Driver
Charging Station
CSMS
User authorization successful.
TransactionEventRequest(eventType = Started, transactionId = AB1234, triggerReason = Authorized, seqNo = N, timestamp, idToken.id = 1234, ...)
TransactionEventResponse(idTokenInfo.status = Accepted,...)
alt
[if within ConnectionTimeOut]
plugin cable
NotifyEventRequest(component.name = Connector, variable.name = AvailabilityState, actualValue = Occupied)
NotifyEventResponse()
TransactionEventRequest(eventType = Updated, transactionId = AB1234, seqNo = N + 1, timestamp, chargingState = EVConnected, triggerReason = CablePluggedIn, ...)
TransactionEventResponse(...)
alt
[if cable not permanently attached]
lock connector
start energy offer
TransactionEventRequest(eventType = Updated, transactionId = AB1234, seqNo = N + 1, timestamp, chargingState = Charging, triggerReason = ChargingStateChanged, ...)
TransactionEventResponse(...)
[if not within Connection Timeout]
TransactionEventRequest(eventType = Ended, triggerReason = EVConnectTimeout, transactionId = AB1234, seqNo = N + 1, timestamp, meterValues, stoppedReason = Timeout)
TransactionEventResponse()
opt
notification

Figure 56. Sequence Diagram: Start Transaction - IdToken First

E03 - Start Transaction - IdToken First - Requirements

Table 65. E03 - Requirements

ID	Precondition	Requirement definition	Note
E03.FR.01	When the IdToken information is known.	The next TransactionEventRequest SHALL contain IdTokenType information.	(Same as E01.FR.13)
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.
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.

ID	Precondition	Requirement definition	Note
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. (Same as E01.FR.07)
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.	(Same as E01.FR.09)
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.	(Same as J02.FR.11)
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*.	(Same as J02.FR.14)
E03.FR.10	`SampledDataSignReadings` is *true*	The Charging Station SHALL retrieve signed meter values and put them in the *signedMeterValue* field of sampledValues.	(Same as E02.FR.14)
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.	(Same as E01.FR.17)
E03.FR.12	When a transaction-related trigger event occurs, listed in *TriggerReasonEnumType* 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 *CablePluggedIn* should be used, not *ChargingStateChanged*. 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. (Same as E02.FR.17)
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 *numberOfPhasesUsed* field.	(Same as E02.FR.18)
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 *numberOfPhasesUsed* field in the next TransactionEventRequest.	(Same as E02.FR.19)
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 `ParkingBayOccupancy`	The Charging Station SHALL deauthorize the transaction and send a TransactionEventRequest (*triggerReason* = `EVConnectionTimeout`) to the CSMS.	Transaction will be ended normally when driver leaves the parking bay.

E04 - Transaction started while Charging Station is offline

No.	Type	Description
1	**Name**	Transaction started while Charging Station is offline
2	**ID**	E04
3	**Objective(s)**	To enable the EV Driver to start a transaction while the Charging Station is *Offline*.
4	**Description**	This use case covers how the Charging Station, while *Offline*, is able to start a transaction using the Local Authorization List or the Authorization Cache.
	Actors	Charging Station, CSMS, EV Driver
	Scenario description	**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.
	Alternative scenario(s)	E10 - Connection Loss During Transaction
5	**Prerequisite(s)**	The Charging Station is *Offline*. The EV Driver is offline/locally authorized by the Charging Station.
6	**Postcondition(s)**	**Successful postcondition:** The TransactionEventRequest has been responded to by the CSMS AND has been removed from the queue of the Charging Station. **Failure postcondition:** The TransactionEventRequest was NOT responded to by the CSMS AND remains in the queue of the Charging Station.
7	**Error handling**	n/a
8	**Remark(s)**	The scenario description and sequence diagram above are not based on a specific `TxStartPoint`. For more details see the use cases: E01 - Start Transaction options.

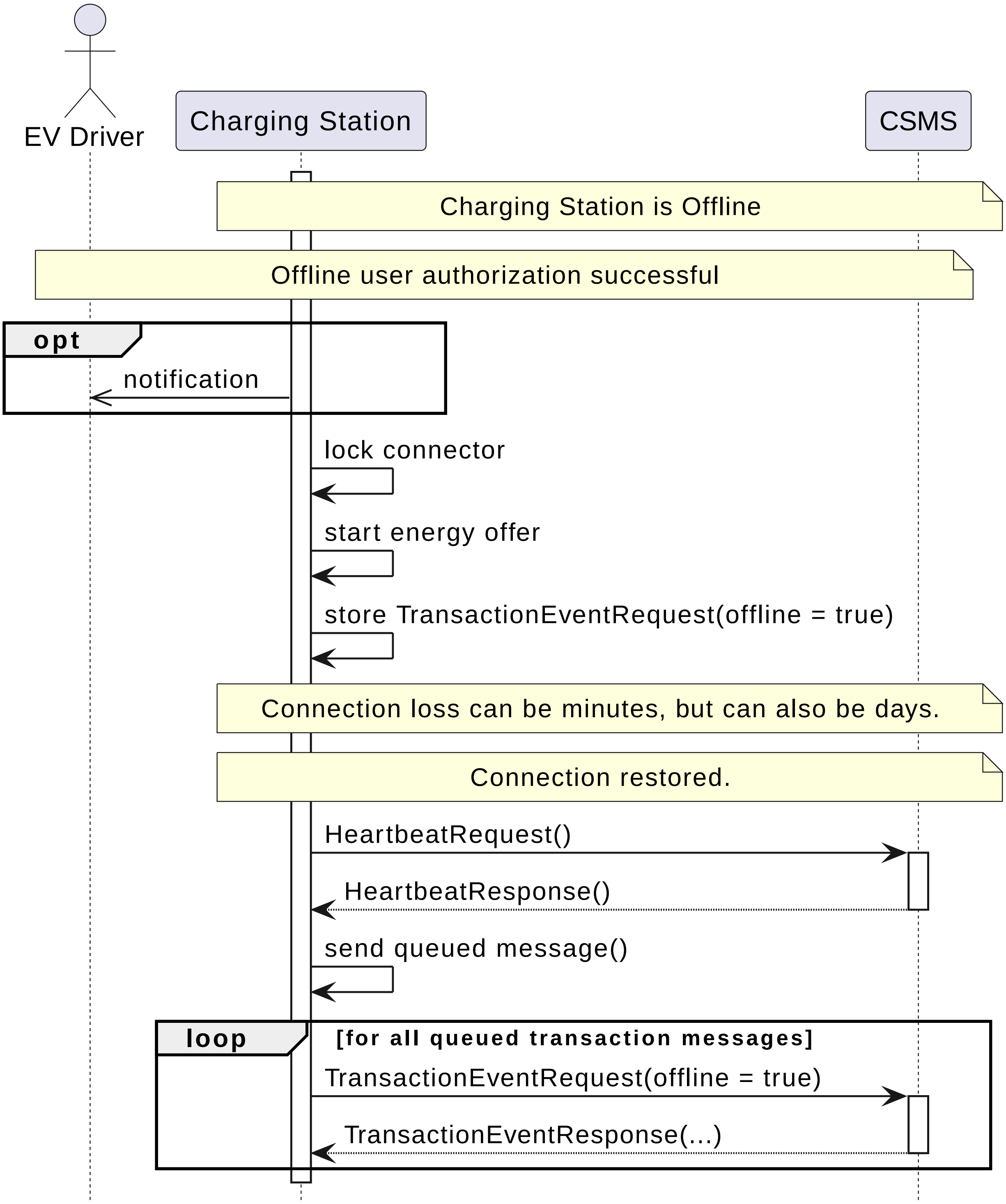

EV Driver
Charging Station
CSMS
Charging Station is Offline
Offline user authorization successful
opt
notification
lock connector
start energy offer
store TransactionEventRequest(offline = true)
Connection loss can be minutes, but can also be days.
Connection restored.
HeartbeatRequest()
HeartbeatResponse()
send queued message()
loop
[for all queued transaction messages]
TransactionEventRequest(offline = true)
TransactionEventResponse(...)

Figure 57. Sequence Diagram: Transaction started while Charging Station is offline

E04 - Transaction started while Charging Station is offline - Requirements

Table 66. E04 - Requirements

ID	Precondition	Requirement definition	Note
E04.FR.01	When *Offline*.	The Charging Station MUST queue any TransactionEventRequest messages.	Same as E12.FR.01
E04.FR.02	After the connection is restored.	The Charging Station MUST send queued TransactionEventRequest messages.	Same as E12.FR.02
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.	Same as E12.FR.02

ID	Precondition	Requirement definition	Note
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. Same E01.FR.07
E04.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 meterValue field with *context* = `Transaction.Begin` in the TransactionEventRequest(eventType = Started) sent to the CSMS to provide more details during the transaction.	Same as E01.FR.09
E04.FR.06	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.	Same as J02.FR.11
E04.FR.07	E04.FR.06 AND *Offline* AND The Charging Station is running low on memory	The Charging Station MAY drop TransactionEventRequest(eventType = Updated) messages.	Same as J02.FR.12
E04.FR.08	E04.FR.07	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.	Same as J02.FR.13
E04.FR.09	E04.FR.06 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*.	Same as J02.FR.14
E04.FR.10	`SampledDataSignReadings` is *true*	The Charging Station SHALL retrieve signed meter values and put them in the *signedMeterValue* field of sampledValues.	Same as E02.FR.14
E04.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.	Same as E01.FR.17

E05 - Start Transaction - Id not Accepted

No.	Type	Description
1	**Name**	Start Transaction - Id not Accepted
2	**ID**	E05
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. Since 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 *idToken*. When receiving a TransactionEventResponse message with *idTokenInfo.status* is not `Accepted`, the Charging Station must stop the energy delivery to the EV.
	Actors	Charging Station, CSMS
	Scenario description	**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 *idTokenInfo.status* that does not allow charging. **3.** The Charging Station suspends the energy offer. (Taking into account: `MaxEnergyOnInvalidId`, if supported) **4. If** `StopTxOnInvalidId` = false, **then** **4a.** The Charging Station sends TransactionEventRequest (eventType = Updated) with *triggerReason* = `ChargingStateChanged` and the *chargingState* = `SuspendedEVSE` and receives TransactionEventResponse from the CSMS. **5. If** `StopTxOnInvalidId` = true, **then** **5a. If** `TxStopPoint` = "EVConnected" the Charging Station sends TransactionEventRequest (eventType = Updated) with *triggerReason* = `Deauthorized` and the *chargingState* = `EVConnected` and receives TransactionEventResponse from the CSMS. **else** **5b.** If `TxStopPoint` = "Authorized" the Charging Station sends TransactionEventRequest (eventType = Ended) with *triggerReason* = `Deauthorized` and the *chargingState* = `EVConnected` 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)**	**Successful postcondition:** The transaction is kept ongoing, and the cable remains locked, but no energy is delivered. **Failure postcondition:** n/a
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`: PowerPathClosed `TxStopPoint`: either EVConnected or Authorized/PowerPathClosed 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. NOTE: The configuration parameter `MaxEnergyOnInvalidId` only plays a role when `StopTxOnInvalidId` is set to *false*.

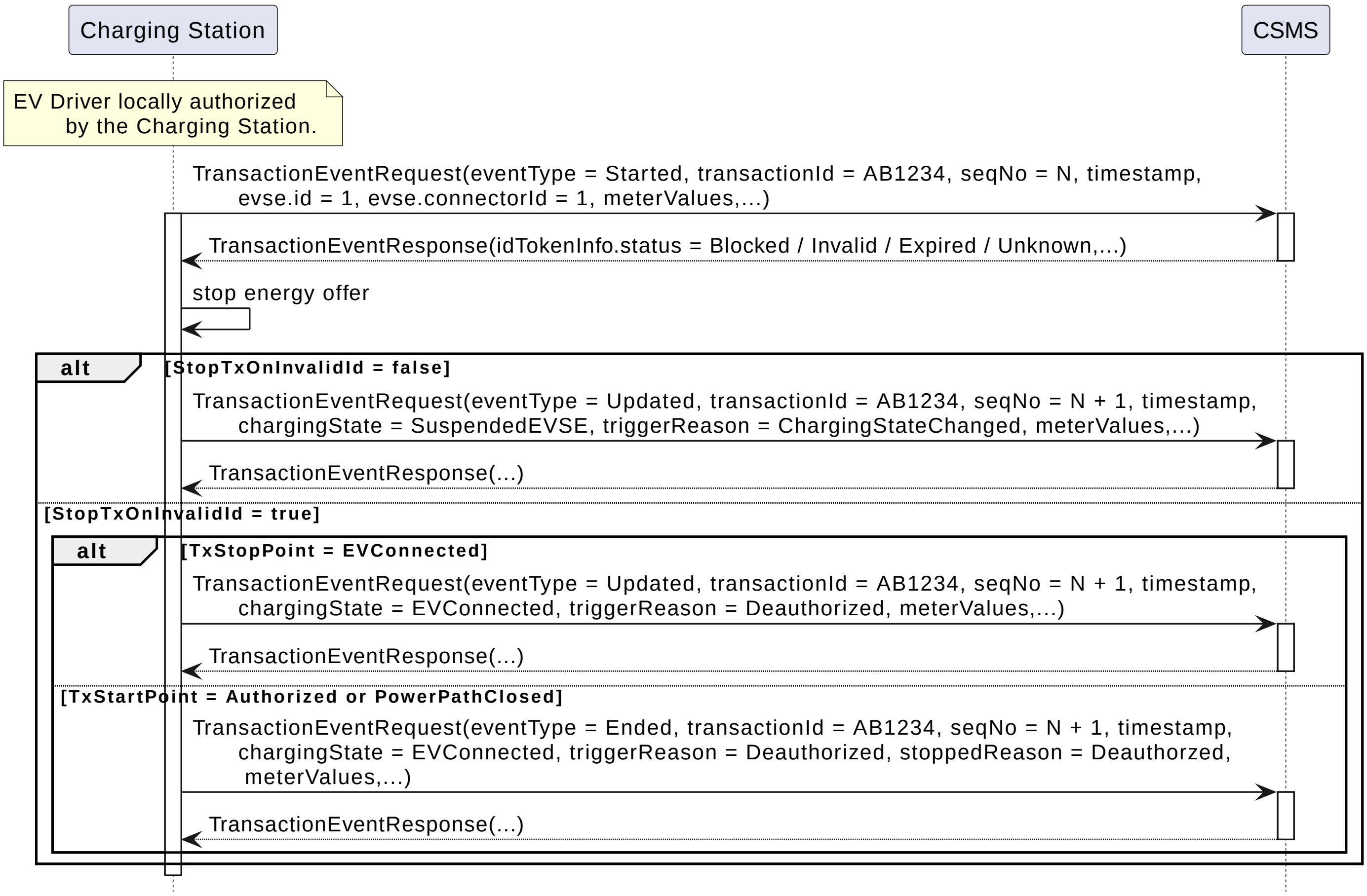

Charging Station
CSMS
EV Driver locally authorized by the Charging Station.
TransactionEventRequest(eventType = Started, transactionId = AB1234, seqNo = N, timestamp, evse.id = 1, evse.connectorId = 1, meterValues,...)
TransactionEventResponse(idTokenInfo.status = Blocked / Invalid / Expired / Unknown,...)
stop energy offer
alt
[StopTxOnInvalidId = false]
TransactionEventRequest(eventType = Updated, transactionId = AB1234, seqNo = N + 1, timestamp, chargingState = SuspendedEVSE, triggerReason = ChargingStateChanged, meterValues,...)
TransactionEventResponse(...)
[StopTxOnInvalidId = true]
alt
[TxStopPoint = EVConnected]
TransactionEventRequest(eventType = Updated, transactionId = AB1234, seqNo = N + 1, timestamp, chargingState = EVConnected, triggerReason = Deauthorized, meterValues,...)
TransactionEventResponse(...)
[TxStartPoint = Authorized or PowerPathClosed]
TransactionEventRequest(eventType = Ended, transactionId = AB1234, seqNo = N + 1, timestamp, chargingState = EVConnected, triggerReason = Deauthorized, stoppedReason = Deauthorzed, meterValues,...)
TransactionEventResponse(...)

Figure 58. Sequence Diagram: Start Transaction - Id not Accepted

E05 - Start Transaction - Id not Accepted - Requirements

Table 67. 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 *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*	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.

ID	Precondition	Requirement definition	Note
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 *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.	
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. Same as E01.FR.07
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 meterValue field with *context* = `Transaction.Begin` in the TransactionEventRequest(eventType = Started) sent to the CSMS to provide more details during the transaction.	Same as E01.FR.09
E05.FR.06	`SampledDataSignReadings` is *true*	The Charging Station SHALL retrieve signed meter values and put them in the *signedMeterValue* field of sampledValues.	Same as E01.FR.14
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 *eventType* = `Started` to the optional meterValue field with *context* = `Transaction.Begin` in the TransactionEventRequest(eventType = Updated) that occurs when charging starts.	Same as E01.FR.17
E05.FR.09 **(2.1)**	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 to `EVConnected`.	If the physical change of charging state in the Charging Station does not occur at (practically) the same time as the trigger `Deauthorized`, then the *chargingState* change may be reported separately as a *triggerReason* = `ChargingStateChanged`. Use of charging state `SuspendedEVSE` in this situation has become deprecated as of OCPP 2.1.
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

No.	Type	Description
1	**Name**	Stop Transaction options
2	**ID**	E06
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 (eventType = Ended), 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	**1.** The Charging Stations parking bay occupancy detector stops detecting the EV. **2.** The Charging Station sends a TransactionEventRequest (eventType = Ended) 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)	**Successful postcondition:** The transaction is ended and the CSMS is *Successfully* informed. **Failure postcondition:** The transaction is still ongoing. *or* The CSMS is *not* informed.

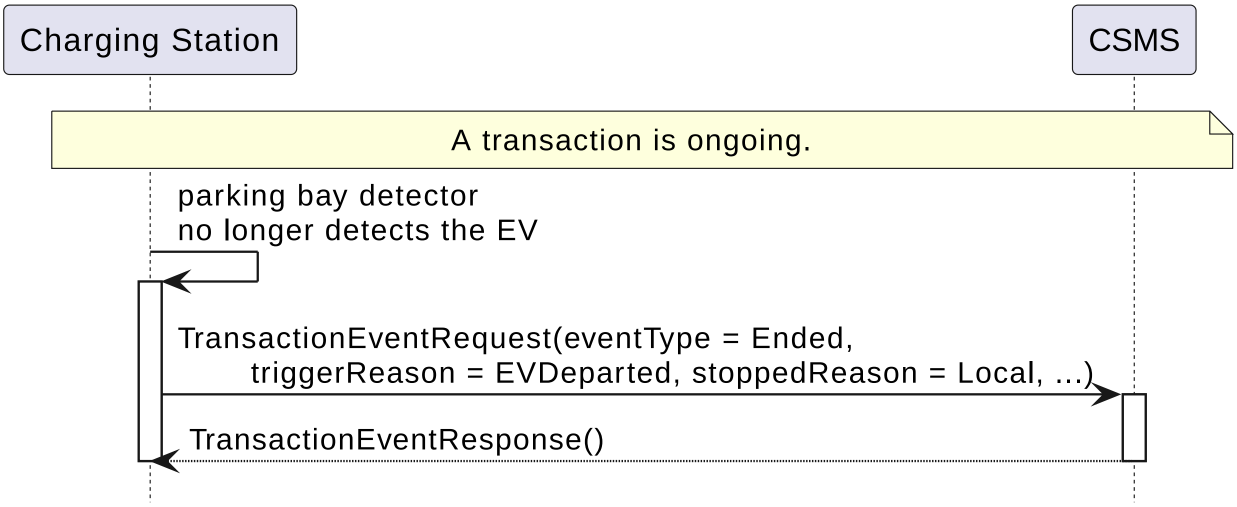

Charging Station
CSMS
A transaction is ongoing.
parking bay detector
no longer detects the EV
TransactionEventRequest(eventType = Ended,
triggerReason = EVDeparted, stoppedReason = Local, ...)
TransactionEventResponse()

Figure 59. 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	**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 NotifyEventRequest with component.name *Connector*, variable.name *AvailabilityState* and actualValue *Available* to the CSMS to inform it about a Connector that became *Available* **3.** The Charging Station sends a TransactionEventRequest (eventType = Ended) 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)	**Successful postcondition:** The transaction is ended and the CSMS is *Successfully* informed. **Failure postcondition:** The transaction is still ongoing. *or* The CSMS is *not* informed.

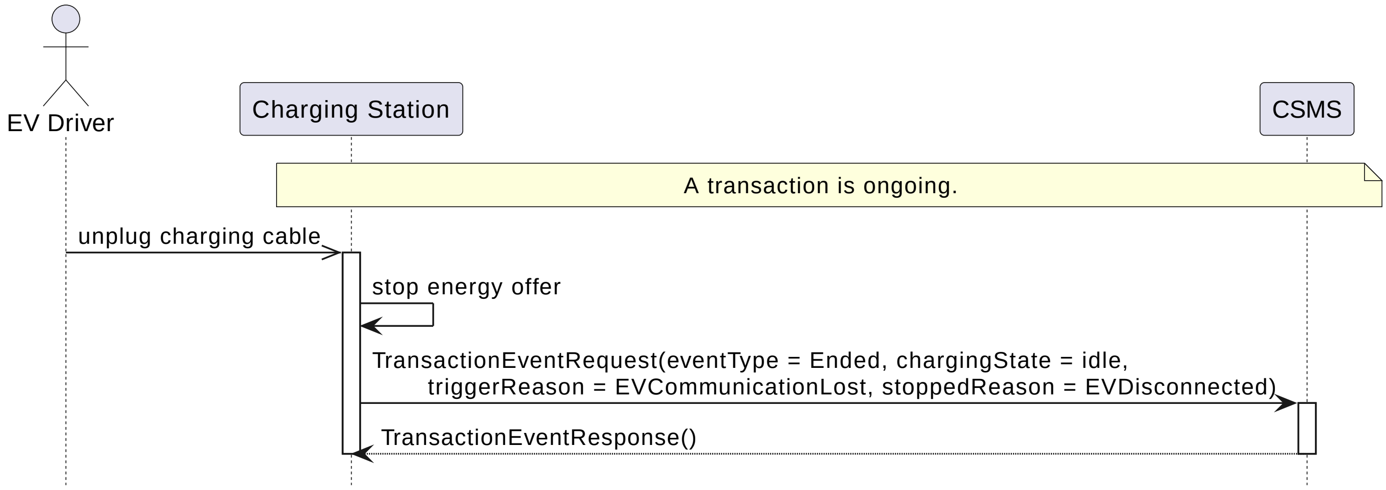

EV Driver
Charging Station
CSMS
A transaction is ongoing.
unplug charging cable
stop energy offer
TransactionEventRequest(eventType = Ended, chargingState = idle,
triggerReason = EVCommunicationLost, stoppedReason = EVDisconnected)
TransactionEventResponse()

Figure 60. Sequence Diagram: Stop Transaction options - EVConnected

S3	*Scenario objective*	**Stop a transaction when the driver is no longer authorized.**
	Scenario description	**1.** The Charging Station sends a TransactionEventRequest to the CSMS. **2.** An invalid IdToken is received in a TransactionEventResponse. **3.** The Charging Station sends a TransactionEventRequest (eventType = Ended) 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: Authorized
	Postcondition(s)	**Successful postcondition:** The transaction is ended and the CSMS is *Successfully* informed. **Failure postcondition:** The transaction is still ongoing. *or* The CSMS is *not* informed.

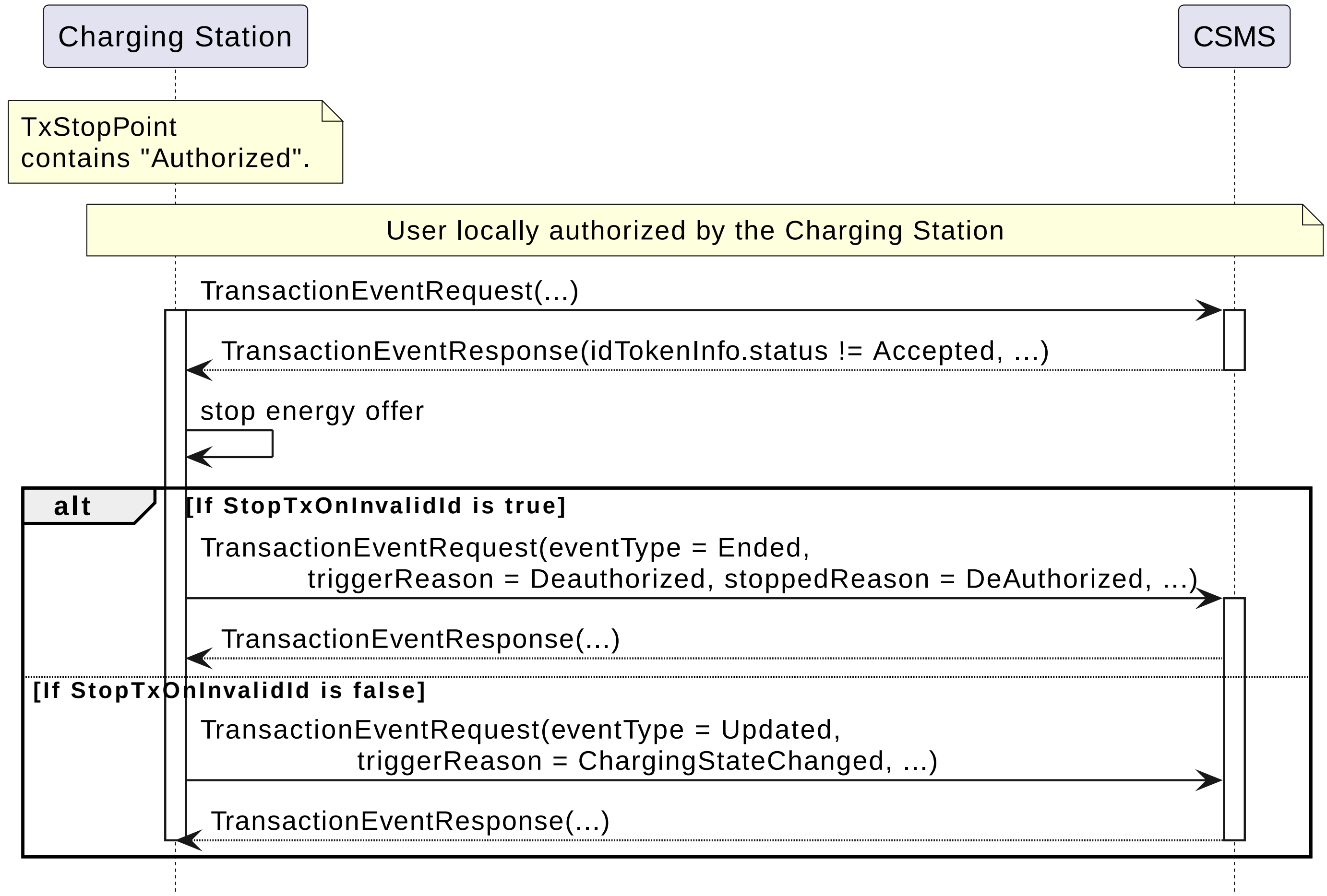

Charging Station
CSMS
TxStopPoint contains "Authorized".
User locally authorized by the Charging Station
TransactionEventRequest(...)
TransactionEventResponse(idTokenInfo.status != Accepted, ...)
stop energy offer
alt
[If StopTxOnInvalidId is true]
TransactionEventRequest(eventType = Ended, triggerReason = Deauthorized, stoppedReason = DeAuthorized, ...)
TransactionEventResponse(...)
[If StopTxOnInvalidId is false]
TransactionEventRequest(eventType = Updated, triggerReason = ChargingStateChanged, ...)
TransactionEventResponse(...)

Figure 61. 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 (eventType = Ended) 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: PowerPathClosed
	Postcondition(s)	**Successful postcondition:** The transaction is ended and the CSMS is *Successfully* informed. **Failure postcondition:** The transaction is still ongoing. *or* The CSMS is *not* informed.

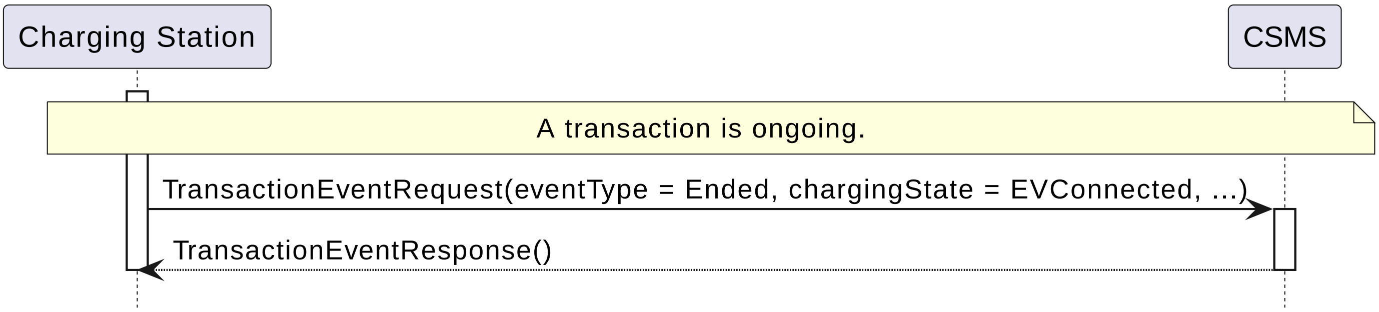

Charging Station
CSMS
A transaction is ongoing.
TransactionEventRequest(eventType = Ended, chargingState = EVConnected, ...)
TransactionEventResponse()

Figure 62. 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	**1.** The energy transfer between EV and the Charging Station stops (for example: EV stops charging). **2.** The Charging Station sends a TransactionEventRequest (eventType = Ended) 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: EnergyTransfer
	Postcondition(s)	**Successful postcondition:** The transaction is ended and the CSMS is *Successfully* informed. **Failure postcondition:** The transaction is still ongoing. *or* The CSMS is *not* informed.

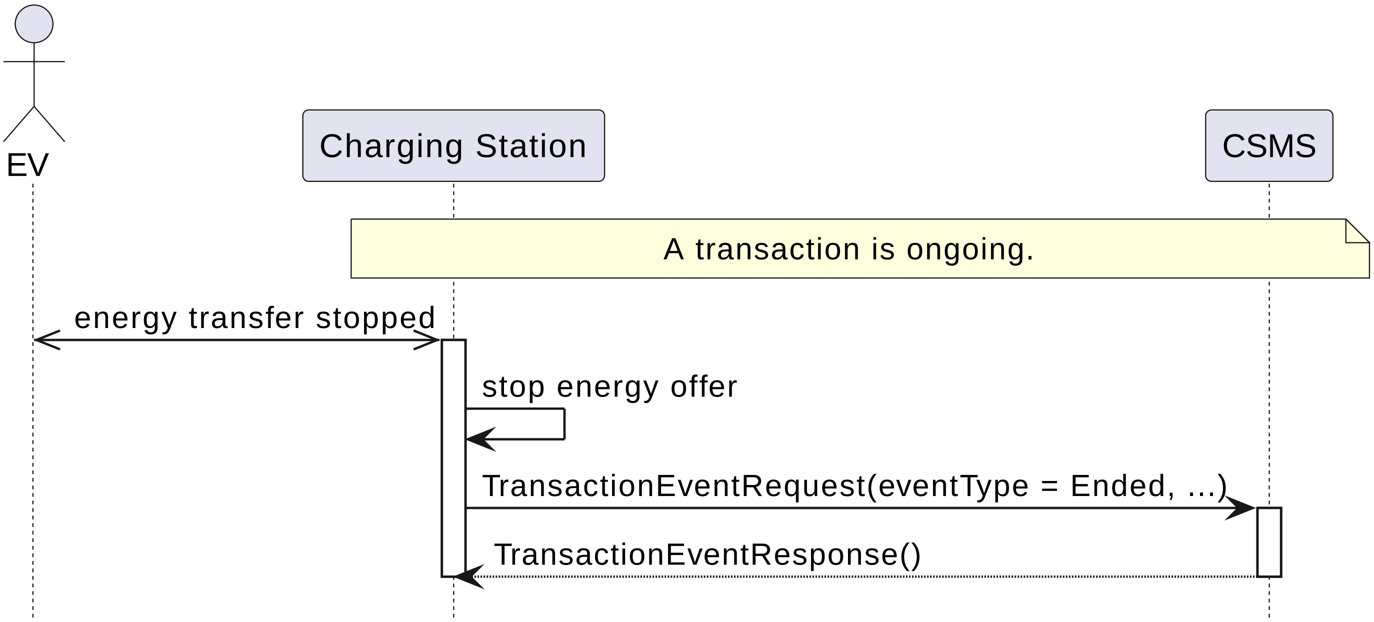

EV
Charging Station
CSMS
A transaction is ongoing.
energy transfer stopped
stop energy offer
TransactionEventRequest(eventType = Ended, ...)
TransactionEventResponse()

Figure 63. Sequence Diagram: Stop Transaction options - EnergyTransfer

S7	*Scenario objective*	**Stop a transaction when EV driver ends authorization**
	Scenario description	**1.** The EV drivers presents an IdToken to end the charging. **2.** The Charging Station sends a TransactionEventRequest (eventType = Ended) 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: Authorized (or PowerPathClosed).
	Postcondition(s)	**Successful postcondition:** The transaction is ended and the CSMS is *Successfully* informed. **Failure postcondition:** The transaction is still ongoing. *or* The CSMS is *not* informed.

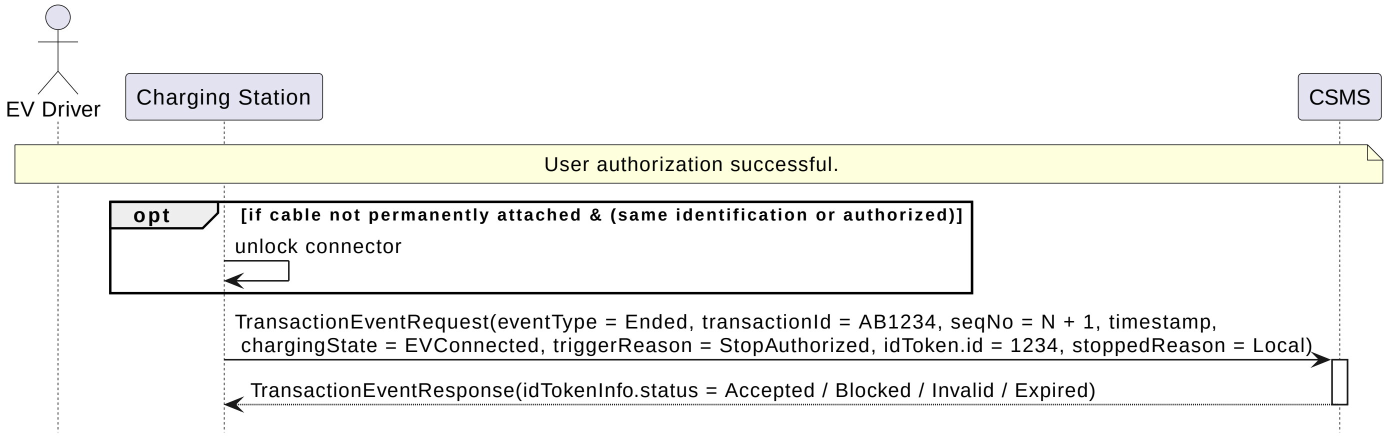

EV Driver
Charging Station
CSMS
User authorization successful.
opt
[if cable not permanently attached & (same identification or authorized)]
unlock connector
TransactionEventRequest(eventType = Ended, transactionId = AB1234, seqNo = N + 1, timestamp, chargingState = EVConnected, triggerReason = StopAuthorized, idToken.id = 1234, stoppedReason = Local)
TransactionEventResponse(idTokenInfo.status = Accepted / Blocked / Invalid / Expired)

Figure 64. Sequence Diagram: Stop Transaction options - Authorized

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

E06 - Stop Transaction options - Requirements

Table 68. 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	See use case E05 - Start Transaction - Id not Accepted,
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. (Same as E01.FR.07)
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. (Same as E01.FR.15)
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

No.	Type	Description
1	**Name**	Transaction locally stopped by IdToken
2	**ID**	E07
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
	Scenario description Reporting StopAuthorized with end of transaction	*TxStopPoint* = `Authorized` (or `PowerPathClosed`) **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.
	Alternative scenario(s) Reporting StopAuthorized in Update event first, then end transaction	*TxStopPoint* = `Authorized` (or `PowerPathClosed`) **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)**	**Successful postcondition:** The CSMS has received all relevant information about the transaction and the Charging Station is in *Idle* status. **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.
7	**Error handling**	n/a
8	**Remark(s)**	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. The CSMS cannot prevent a transaction from stopping.

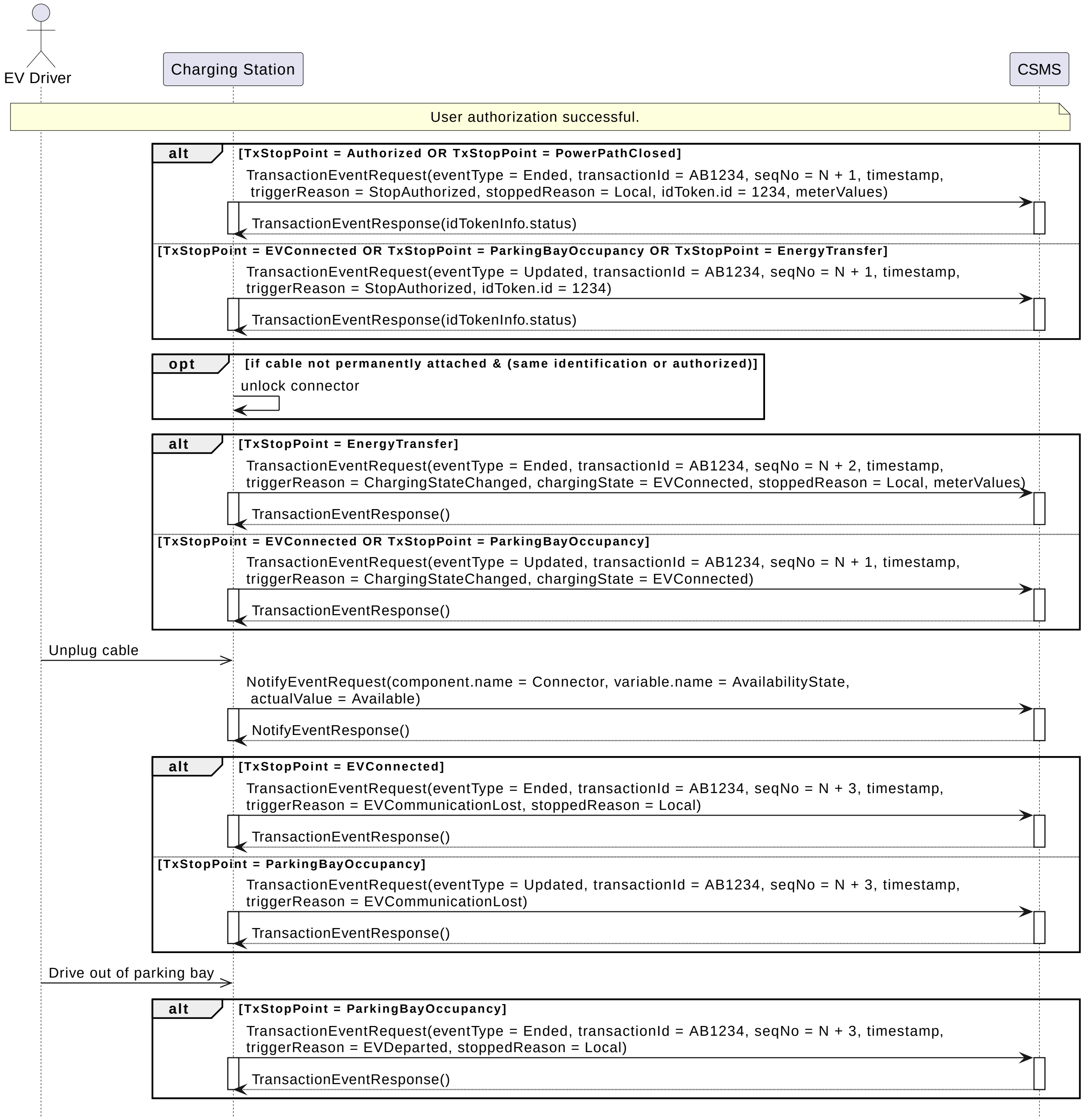

EV Driver
Charging Station
CSMS
User authorization successful.
alt
[TxStopPoint = Authorized OR TxStopPoint = PowerPathClosed]
TransactionEventRequest(eventType = Ended, transactionId = AB1234, seqNo = N + 1, timestamp, triggerReason = StopAuthorized, stoppedReason = Local, idToken.id = 1234, meterValues)
TransactionEventResponse(idTokenInfo.status)
[TxStopPoint = EVConnected OR TxStopPoint = ParkingBayOccupancy OR TxStopPoint = EnergyTransfer]
TransactionEventRequest(eventType = Updated, transactionId = AB1234, seqNo = N + 1, timestamp, triggerReason = StopAuthorized, idToken.id = 1234)
TransactionEventResponse(idTokenInfo.status)
opt
[if cable not permanently attached & (same identification or authorized)]
unlock connector
alt
[TxStopPoint = EnergyTransfer]
TransactionEventRequest(eventType = Ended, transactionId = AB1234, seqNo = N + 2, timestamp, triggerReason = ChargingStateChanged, chargingState = EVConnected, stoppedReason = Local, meterValues)
TransactionEventResponse()
[TxStopPoint = EVConnected OR TxStopPoint = ParkingBayOccupancy]
TransactionEventRequest(eventType = Updated, transactionId = AB1234, seqNo = N + 1, timestamp, triggerReason = ChargingStateChanged, chargingState = EVConnected)
TransactionEventResponse()
Unplug cable
NotifyEventRequest(component.name = Connector, variable.name = AvailabilityState, actualValue = Available)
NotifyEventResponse()
alt
[TxStopPoint = EVConnected]
TransactionEventRequest(eventType = Ended, transactionId = AB1234, seqNo = N + 3, timestamp, triggerReason = EVCommunicationLost, stoppedReason = Local)
TransactionEventResponse()
[TxStopPoint = ParkingBayOccupancy]
TransactionEventRequest(eventType = Updated, transactionId = AB1234, seqNo = N + 3, timestamp, triggerReason = EVCommunicationLost)
TransactionEventResponse()
Drive out of parking bay
alt
[TxStopPoint = ParkingBayOccupancy]
TransactionEventRequest(eventType = Ended, transactionId = AB1234, seqNo = N + 3, timestamp, triggerReason = EVDeparted, stoppedReason = Local)
TransactionEventResponse()

Figure 65. Sequence Diagram: Transaction locally stopped by IdToken with TransactionEventRequest reported strictly by TxStopPoint configuration

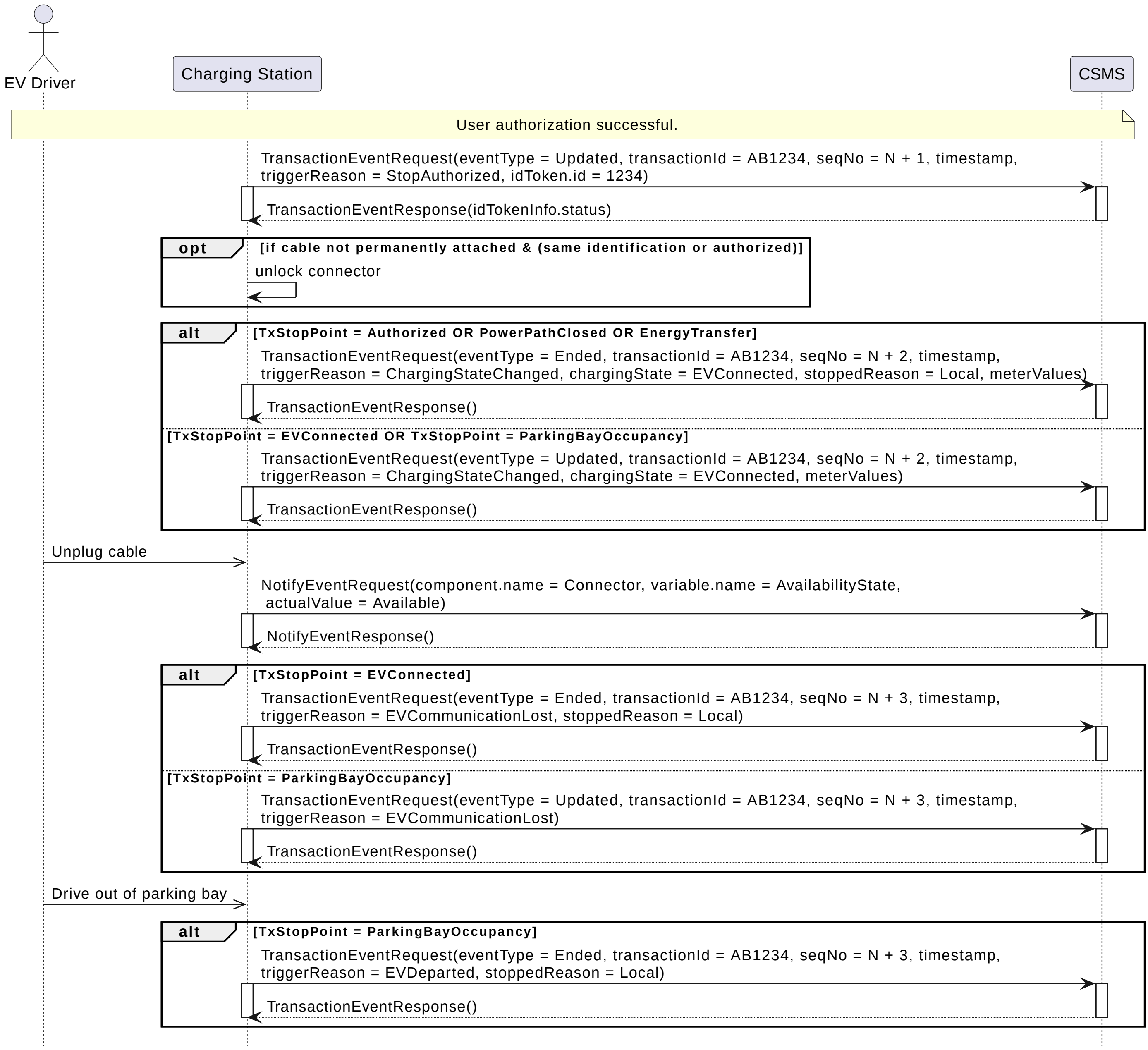

EV Driver
Charging Station
CSMS
User authorization successful.
TransactionEventRequest(eventType = Updated, transactionId = AB1234, seqNo = N + 1, timestamp, triggerReason = StopAuthorized, idToken.id = 1234)
TransactionEventResponse(idTokenInfo.status)
opt
[if cable not permanently attached & (same identification or authorized)]
unlock connector
alt
[TxStopPoint = Authorized OR PowerPathClosed OR EnergyTransfer]
TransactionEventRequest(eventType = Ended, transactionId = AB1234, seqNo = N + 2, timestamp, triggerReason = ChargingStateChanged, chargingState = EVConnected, stoppedReason = Local, meterValues)
TransactionEventResponse()
[TxStopPoint = EVConnected OR TxStopPoint = ParkingBayOccupancy]
TransactionEventRequest(eventType = Updated, transactionId = AB1234, seqNo = N + 2, timestamp, triggerReason = ChargingStateChanged, chargingState = EVConnected, meterValues)
TransactionEventResponse()
Unplug cable
NotifyEventRequest(component.name = Connector, variable.name = AvailabilityState, actualValue = Available)
NotifyEventResponse()
alt
[TxStopPoint = EVConnected]
TransactionEventRequest(eventType = Ended, transactionId = AB1234, seqNo = N + 3, timestamp, triggerReason = EVCommunicationLost, stoppedReason = Local)
TransactionEventResponse()
[TxStopPoint = ParkingBayOccupancy]
TransactionEventRequest(eventType = Updated, transactionId = AB1234, seqNo = N + 3, timestamp, triggerReason = EVCommunicationLost)
TransactionEventResponse()
Drive out of parking bay
alt
[TxStopPoint = ParkingBayOccupancy]
TransactionEventRequest(eventType = Ended, transactionId = AB1234, seqNo = N + 3, timestamp, triggerReason = EVDeparted, stoppedReason = Local)
TransactionEventResponse()

Figure 66. Sequence Diagram: Transaction locally stopped by IdToken with delayed TransactionEventRequest eventType = Ended for TxStopPoint = Authorized OR PowerPathClosed

E07 - Transaction locally stopped by IdToken - Requirements

Table 69. 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)

ID	Precondition	Requirement definition	Note
E07.FR.02	E07.FR.01	The Charging Station SHALL send a TransactionEventRequest with *triggerReason* = `StopAuthorized` and SHOULD include the *idToken* used to stop authorization.	The stopping *idToken* may differ from the starting *idToken*, when they share the same GroupId.
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.
E07.FR.05	If a transaction is stopped on request of the EV driver at the Charging Station.	Charging Station SHOULD use a stoppedReason = `Local` in 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.
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 ReasonEnumType for *stoppedReason* in the final TransactionEventRequest with *eventType* = `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.
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 (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.	Same as E06.FR.11
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(eventType = Ended) message.	Same as E06.FR.12
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.	Same as E06.FR.13
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. Same as E01.FR.07
E07.FR.12	`SampledDataSignReadings` is *true*	The Charging Station SHALL retrieve signed meter values and put them in the *signedMeterValue* field of sampledValues.	Same as E02.FR.14

E08 - Transaction stopped while Charging Station is offline

No.	Type	Description
1	**Name**	Transaction stopped while Charging Station is offline
2	**ID**	E08
	Parent use case	E07 - Local Stop Transaction
3	**Objective(s)**	To enable the EV Driver to stop a transaction while the Charging Station is *Offline*.
4	**Description**	This use case describes how an EV Driver can stop a transaction while the Charging Station is *Offline*. While a transaction is ongoing and the Charging Station is *Offline*, 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 *Offline* stop transaction to the CSMS.
	Actors	Charging Station, CSMS, EV Driver
	Scenario description	**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 *Idle* status.
7	**Error handling**	n/a
8	**Remark(s)**	groupId check must be done on Local Authorization List and / or Authorization Cache if available. 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 message are sent. For more details see the use case: E06 - Stop Transaction options

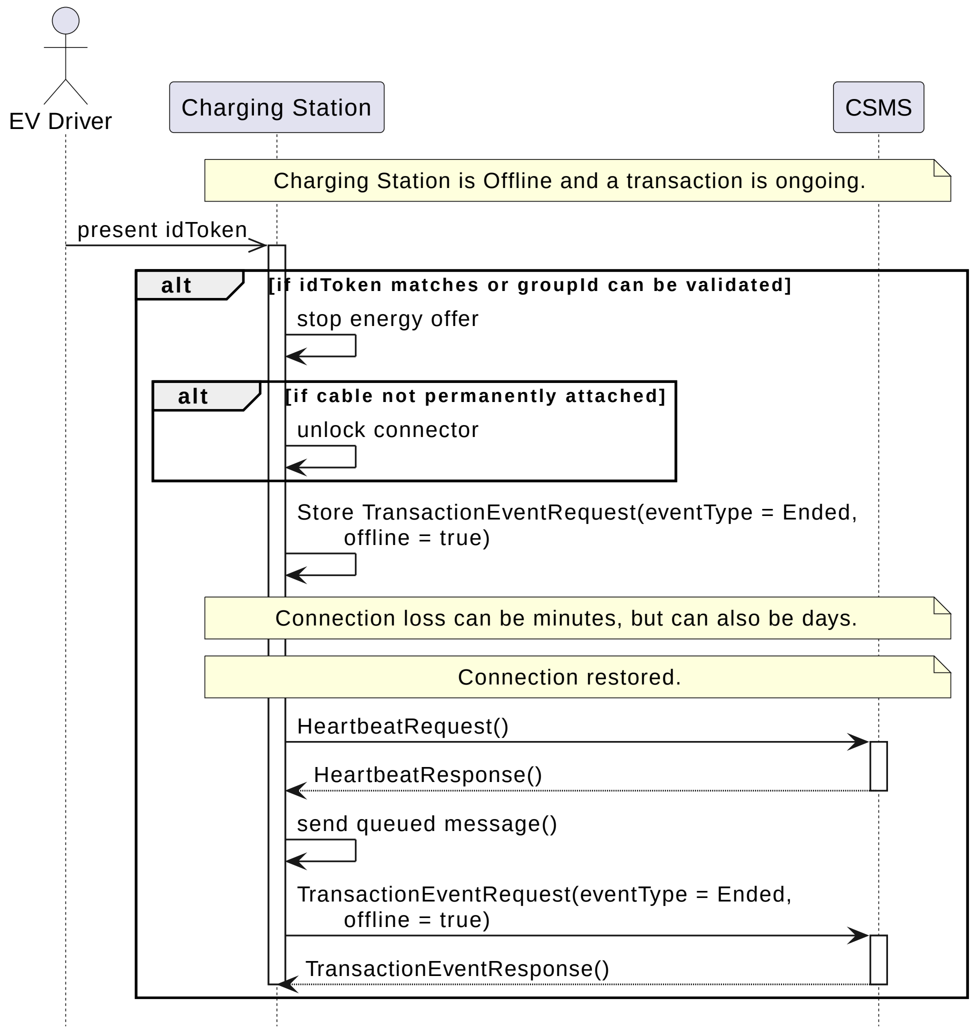

EV Driver
Charging Station
CSMS
Charging Station is Offline and a transaction is ongoing.
present idToken
alt
[if idToken matches or groupId can be validated]
stop energy offer
alt
[if cable not permanently attached]
unlock connector
Store TransactionEventRequest(eventType = Ended,
offline = true)
Connection loss can be minutes, but can also be days.
Connection restored.
HeartbeatRequest()
HeartbeatResponse()
send queued message()
TransactionEventRequest(eventType = Ended,
offline = true)
TransactionEventResponse()

Figure 67. Sequence Diagram: Transaction stopped while Charging Station is offline

E08 - Transaction stopped while Charging Station is offline - Requirements

Table 70. 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).	
E08.FR.05	When *Offline*.	The Charging Station MUST queue any TransactionEventRequest messages.	Same as E12.FR.01
E08.FR.06	After the connection is restored.	The Charging Station MUST send queued TransactionEventRequest messages.	E12.FR.02
E08.FR.07		The flag: *offline* SHALL be set to TRUE for any TransactionEventRequest that occurred while the Charging Station was offline.	E12.FR.02
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. Same E01.FR.07

ID	Precondition	Requirement definition	Note
E08.FR.09	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.	
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(eventType = Ended) message.	Same as E06.FR.12
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.	Same as E06.FR.13
E08.FR.12	`SampledDataSignReadings` is *true*	The Charging Station SHALL retrieve signed meter values and put them in the *signedMeterValue* field of sampledValues.	Same as E02.FR.14

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
	Parent use case	E07 - Local Stop Transaction
3	**Objective(s)**	To stop an ongoing transaction when the Charging Cable is unplugged on the EV side.
4	**Description**	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: `StopTxOnEVSideDisconnect` = true. 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 TransactionEventRequest to the CSMS. The Charging Cable, if locked and `UnlockOnEvSideDisconnect` = false, will remain locked at the Charging Station until the EV Driver returns and presents his/hers IdToken. Otherwise it will unlock the cable.
	Actors	Charging Station, CSMS, EV Driver
	Scenario description	**1.** The cable is unplugged at the EV. **2.** The energy offer is suspended. **3.** The Charging Station sends TransactionEventRequest (eventType = Ended, stoppedReason = EVDisconnected) to the CSMS. **4.** The CSMS responds with TransactionEventResponse. **5.** The EV Driver is authorized and unplugs the cable. **6.** The Charging Station sends NotifyEventRequest with component.name *Connector*, variable.name *AvailabilityState* and actualValue *Available* to the CSMS. **7.** The CSMS responds with NotifyEventResponse.
	Alternative scenario(s)	E10 - When cable disconnected on EV-side: Suspend Transaction
5	**Prerequisite(s)**	Configuration Variable: `StopTxOnEVSideDisconnect` = true A transaction is ongoing
6	**Postcondition(s)**	**Successful postcondition:** The Charging Station is in *Idle* status. **Failure postcondition:** n/a
7	**Error handling**	n/a
8	**Remark(s)**	When the Charging Cable is plugged back in, the charging will not resume/continue. The scenario description and sequence diagram above are based on the Configuration Variable for stop transaction being configured as follows. `TxStopPoint`: 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 message are sent. For more details see the use case: E06 - Stop Transaction options

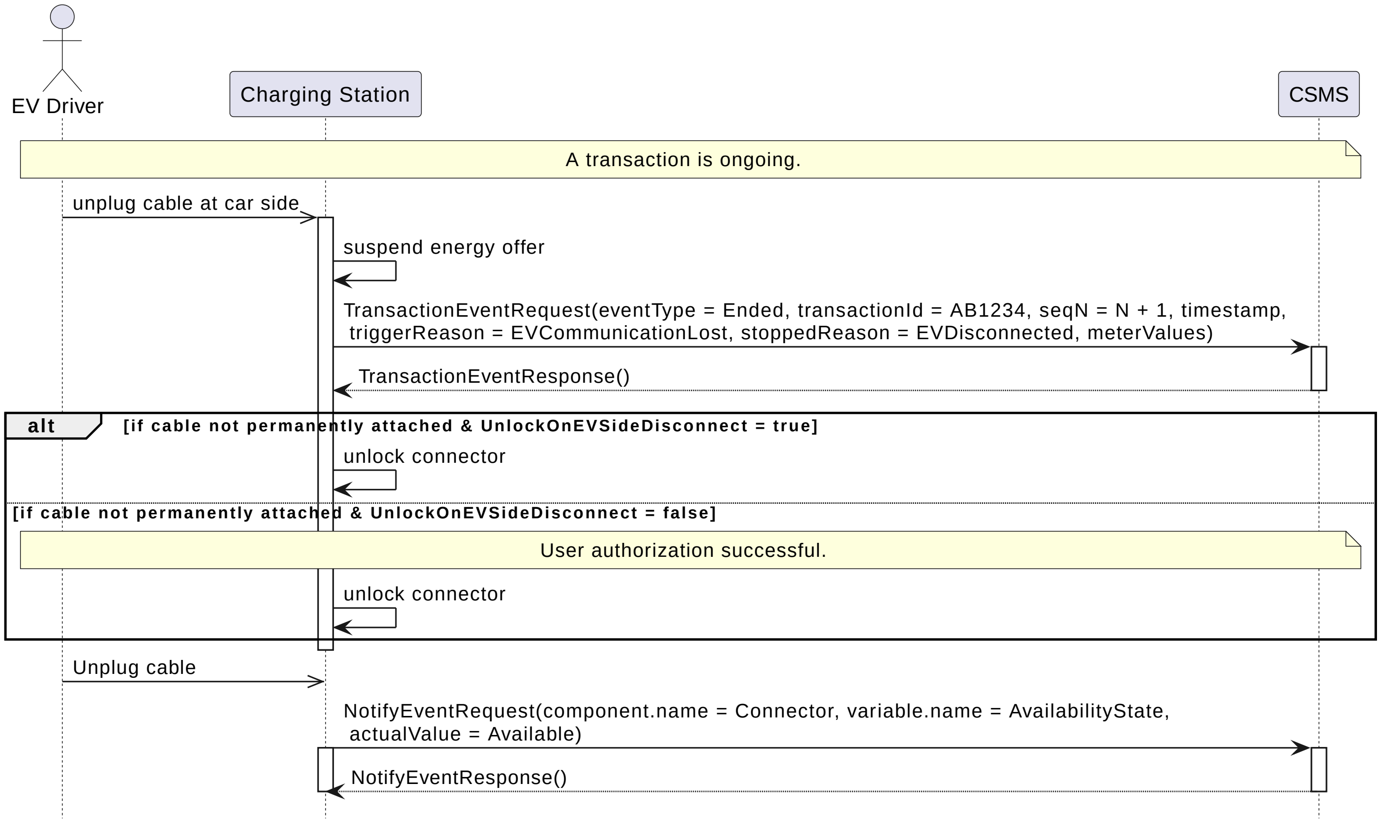

EV Driver
Charging Station
CSMS
A transaction is ongoing.
unplug cable at car side
suspend energy offer
TransactionEventRequest(eventType = Ended, transactionId = AB1234, seqN = N + 1, timestamp, triggerReason = EVCommunicationLost, stoppedReason = EVDisconnected, meterValues)
TransactionEventResponse()
alt
[if cable not permanently attached & UnlockOnEVSideDisconnect = true]
unlock connector
[if cable not permanently attached & UnlockOnEVSideDisconnect = false]
User authorization successful.
unlock connector
Unplug cable
NotifyEventRequest(component.name = Connector, variable.name = AvailabilityState, actualValue = Available)
NotifyEventResponse()

Figure 68. Sequence Diagram: When cable disconnected on EV-side: Stop Transaction

E09 - When cable disconnected on EV-side: Stop Transaction - Requirements

Table 71. E09 - Requirements

ID	Precondition	Requirement definition	Note
E09.FR.01	If `StopTxOnEVSideDisconnect` = *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 `StopTxOnEVSideDisconnect` 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 `UnlockOnEvSideDisconnect` = true.	The Charging Station SHALL unlock the Charging Cable.	
E09.FR.03	E09.FR.01 AND the cable is not permanently attached AND `UnlockOnEvSideDisconnect` = false.	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. Same as E01.FR.07
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.	Same as E08.FR.09

ID	Precondition	Requirement definition	Note
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.	Same as E06.FR.12
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.	Same as E06.FR.13
E09.FR.08	`SampledDataSignReadings` is *true*	The Charging Station SHALL retrieve signed meter values and put them in the *signedMeterValue* field of sampledValues.	Same as E02.FR.14

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
	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**	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: `StopTxOnEVSideDisconnect` = false. 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.
	Actors	Charging Station, CSMS, EV Driver
	Scenario description	**1.** EV Driver unplugs the cable at the EV while a transaction is ongoing. **2.** The energy offer is suspended. *When the EV Driver plugs the cable back in, the transaction is resumed.* **A1.** The Charging Station sends a TransactionEventRequest (eventType = Updated, trigger = CablePluggedIn) **A2.** The CSMS responds with a TransactionEventResponse. *If cable not permanently attached* **B1.** The EV Driver is authorized by the Charging Station and/or CSMS to unlock the charging cable. **B2.** The cable is unlocked. **B3.** The Charging Station sends a TransactionEventRequest (eventType = Ended, trigger = StopAuthorized). **B4.** The EV Driver removes the charging cable. **B5.** The Charging Station sends a NotifyEventRequest with component.name *Connector*, variable.name *AvailabilityState* and actualValue *Available* to the CSMS. **B6.** The CSMS responds with a NotifyEventResponse. *If cable permanently attached* **C1.** The cable is not plugged in within timeout. **C2.** The Charging Station sends a TransactionEventRequest (eventType = Ended, trigger = EVCommunicationLost, stoppedReason = EVDisconnected). **C3.** The Charging Station sends a NotifyEventRequest with component.name *Connector*, variable.name *AvailabilityState* and actualValue *Available* to the CSMS. **C4.** The CSMS responds with a NotifyEventResponse.
	Alternative scenario(s)	E09 - When cable disconnected on EV-side: Stop Transaction
5	**Prerequisite(s)**	Configuration Variable: `StopTxOnEVSideDisconnect` = false A transaction is ongoing
6	**Postcondition(s)**	**Successful postcondition:** The Charging Station is in *Idle* status. The regular transaction is resumed. **Failure postcondition:** n/a
7	**Error handling**	n/a

No.	Type	Description
8	**Remark(s)**	When the Charging Cable is plugged back in, the charging is resumed. 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. The scenario description and sequence diagram above are based on the Configuration Variable for stop transaction being configured as follows. `TxStopPoint`: ParkingBayOccupancy, 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 message are sent. For more details see the use case: E06 - Stop Transaction options

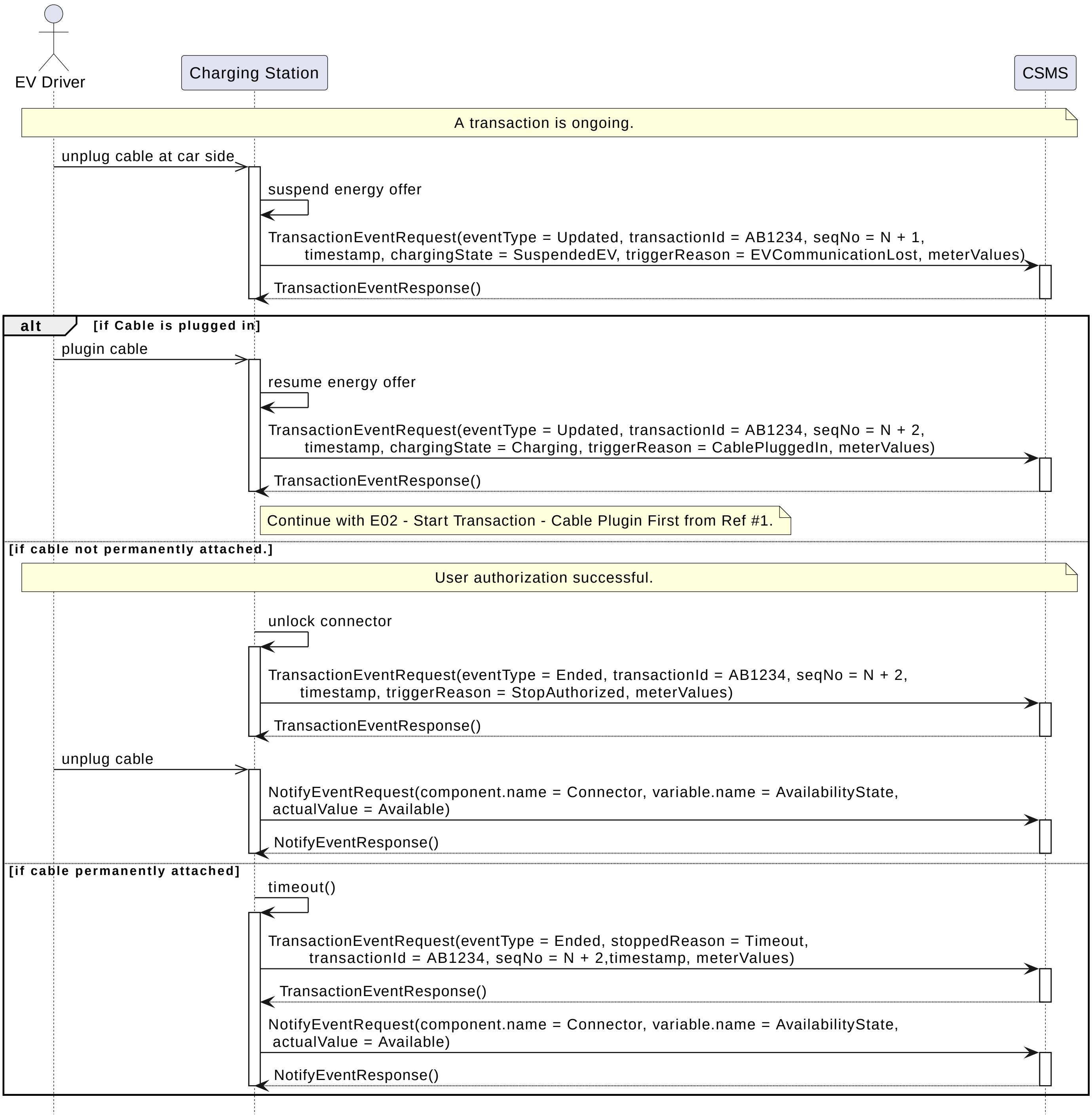

EV Driver
Charging Station
CSMS
A transaction is ongoing.
unplug cable at car side
suspend energy offer
TransactionEventRequest(eventType = Updated, transactionId = AB1234, seqNo = N + 1, timestamp, chargingState = SuspendedEV, triggerReason = EVCommunicationLost, meterValues)
TransactionEventResponse()
alt
[if Cable is plugged in]
plugin cable
resume energy offer
TransactionEventRequest(eventType = Updated, transactionId = AB1234, seqNo = N + 2, timestamp, chargingState = Charging, triggerReason = CablePluggedIn, meterValues)
TransactionEventResponse()
Continue with E02 - Start Transaction - Cable Plugin First from Ref #1.
[if cable not permanently attached.]
User authorization successful.
unlock connector
TransactionEventRequest(eventType = Ended, transactionId = AB1234, seqNo = N + 2, timestamp, triggerReason = StopAuthorized, meterValues)
TransactionEventResponse()
unplug cable
NotifyEventRequest(component.name = Connector, variable.name = AvailabilityState, actualValue = Available)
NotifyEventResponse()
[if cable permanently attached]
timeout()
TransactionEventRequest(eventType = Ended, stoppedReason = Timeout, transactionId = AB1234, seqNo = N + 2,timestamp, meterValues)
TransactionEventResponse()
NotifyEventRequest(component.name = Connector, variable.name = AvailabilityState, actualValue = Available)
NotifyEventResponse()

Figure 69. Sequence Diagram: When cable disconnected on EV-side: Suspend Transaction

E10 - When cable disconnected on EV-side: Suspend Transaction - Requirements

Table 72. 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. Same as E01.FR.07

ID	Precondition	Requirement definition	Note
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.	Same as E08.FR.09
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.	Same as E06.FR.12
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.	Same as E06.FR.13
E10.FR.07	`SampledDataSignReadings` is *true*	The Charging Station SHALL retrieve signed meter values and put them in the *signedMeterValue* field of sampledValues.	Same as E02.FR.14

E11 - Connection Loss During Transaction

No.	Type	Description
1	**Name**	Connection Loss During Transaction
2	**ID**	E11
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	**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)**	**Successful postcondition:** The Charging Station resumes regular communication. **Failure postcondition:** n/a
7	**Error handling**	n/a
8	**Remark(s)**	n/a

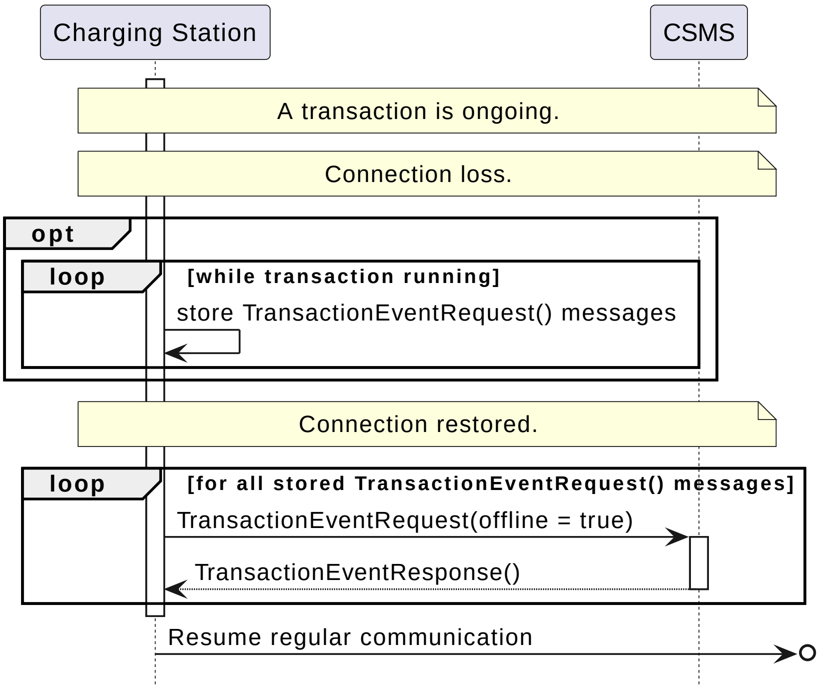

Charging Station
CSMS
A transaction is ongoing.
Connection loss.
opt
loop
[while transaction running]
store TransactionEventRequest() messages
Connection restored.
loop
[for all stored TransactionEventRequest() messages]
TransactionEventRequest(offline = true)
TransactionEventResponse()
Resume regular communication

Figure 70. Sequence Diagram: Connection Loss During Transaction

E11 - Connection Loss During Transaction - Requirements

Table 73. 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. (Same as E12.FR.01)

ID	Precondition	Requirement definition
E11.FR.02	After the connection is restored.	The Charging Station MUST send queued TransactionEventRequest messages with the flag *offline* set to TRUE. (Same as E12.FR.02)
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 meterValue field in the TransactionEventRequest (eventType = Updated) sent to the CSMS to provide more details during the transaction. (Same as J02.FR.11)
E11.FR.04	E11.FR.03 AND *Offline* AND The Charging Station is running low on memory	The Charging Station MAY drop TransactionEventRequest (eventType = Updated) messages. (Same as J02.FR.12)
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. (Same as J02.FR.13)
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 *timestamp*. (Same as J02.FR.14)
E11.FR.07		If the Charging Station goes offline, every message that is still in the queue SHALL be set *Offline*.
E11.FR.08	`SampledDataSignReadings` is *true*	The Charging Station SHALL retrieve signed meter values and put them in the *signedMeterValue* field of sampledValues. (Same as E02.FR.14)

E12 - Inform CSMS of an Offline Occurred Transaction

No.	Type	Description
1	**Name**	Inform CSMS of an Offline Occurred Transaction
2	**ID**	E12
3	**Objective(s)**	To enable the Charging Station to inform the CSMS that a transaction occurred while the Charging Station was *Offline*.
4	**Description**	This use case covers how the Charging Station starts and stops a transaction since connection loss.
	Actors	Charging Station, CSMS
	Scenario description	**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 (eventType = Started, offline = TRUE) to the CSMS. **4.** The CSMS responds with TransactionEventResponse, accepting the transaction. **5.** The Charging Station sends TransactionEventRequest (eventType = Updated, offline = TRUE) **6.** The CSMS responds with TransactionEventResponse. **7.** The Charging Station sends TransactionEventRequest (eventType = Ended, offline = TRUE) **8.** The CSMS responds with TransactionEventResponse.
5	**Prerequisite(s)**	At least one *Offline* transaction has taken place.
6	**Postcondition(s)**	**Successful postcondition:** The CSMS has processed all transactions that occurred *Offline*. **Failure postcondition:** n/a
7	**Error handling**	n/a
8	**Remark(s)**	n/a

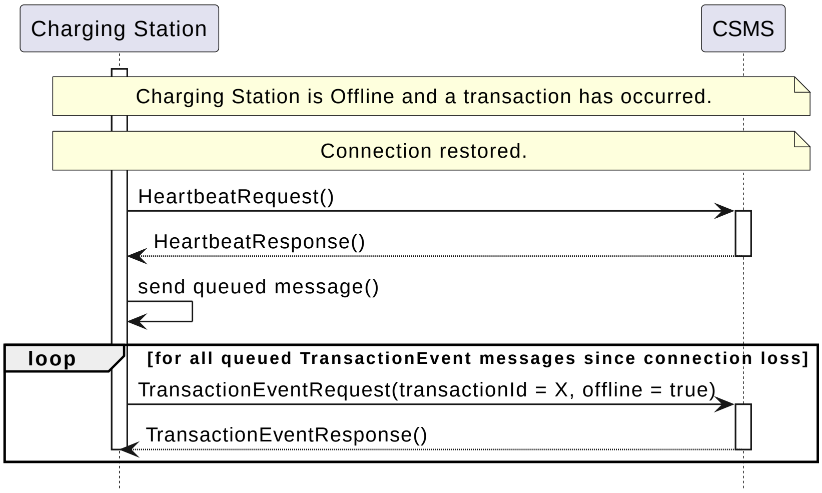

Charging Station
CSMS
Charging Station is Offline and a transaction has occurred.
Connection restored.
HeartbeatRequest()
HeartbeatResponse()
send queued message()
loop
[for all queued TransactionEvent messages since connection loss]
TransactionEventRequest(transactionId = X, offline = true)
TransactionEventResponse()

Figure 71. Sequence Diagram: Inform CSMS of an Offline Occurred Transaction

E12 - Inform CSMS of an Offline Occurred Transaction - Requirements

Table 74. 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.
E12.FR.02	After the connection is restored.	The Charging Station MUST send queued TransactionEventRequest messages with the flag *offline* set to TRUE.

ID	Precondition	Requirement definition
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 meterValue field in the TransactionEventRequest (eventType = Updated) sent to the CSMS to provide more details during the transaction. (Same as J02.FR.11)
E12.FR.04 **(2.1)**	E12.FR.03 AND Charging station is offline AND The Charging Station is running low on memory	The Charging Station MAY drop TransactionEventRequest (eventType = Updated) messages that do not have *triggerReason* `LimitSet`, `CostLimitReached`, `EnergyLimitReached`, `TimeLimitReached` or `SoCLimitReached`.
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. (Same as J02.FR.13)
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 *timestamp*. (Same as J02.FR.14)
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 meterValue field in the TransactionEventRequest (eventType = Ended) sent to the CSMS to provide more details about transaction usage. (Same as E08.FR.09)
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. (Same as E06.FR.12)
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. (Same as E06.FR.13)
E12.FR.10	`SampledDataSignReadings` is *true*	The Charging Station SHALL retrieve signed meter values and put them in the *signedMeterValue* field of sampledValues. (Same as E06.FR.14)

E13 - Transaction-related message not accepted by CSMS

No.	Type	Description
1	**Name**	Transaction-related message not accepted by CSMS
2	**ID**	E13
3	**Objective(s)**	To define how a Charging Station shall handle not accepted messages.
4	**Description**	There are a 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	**1.** The Charging Station sends a transaction-related message to the CSMS. **2.** The message is not accepted and `MessageAttemptsTransactionEvent` not reached. **3.** The Charging Station waits the number of preceding transmissions of this same message times `MessageAttemptIntervalTransactionEvent` seconds. **4.** The Charging Station resends the transaction-related message to the CSMS.
5	**Prerequisite(s)**	n/a
6	**Postcondition(s)**	**Successful postcondition:** MessageAttemptsTransactionEvent is *not* reached AND the transaction-related message is accepted. MessageAttemptsTransactionEvent is reached AND the transaction-related message is disposed. **Failure postcondition:** MessageAttemptsTransactionEvent is *not* reached AND the transaction-related message is disposed. MessageAttemptsTransactionEvent is reached AND the transaction-related message is accepted.
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*.

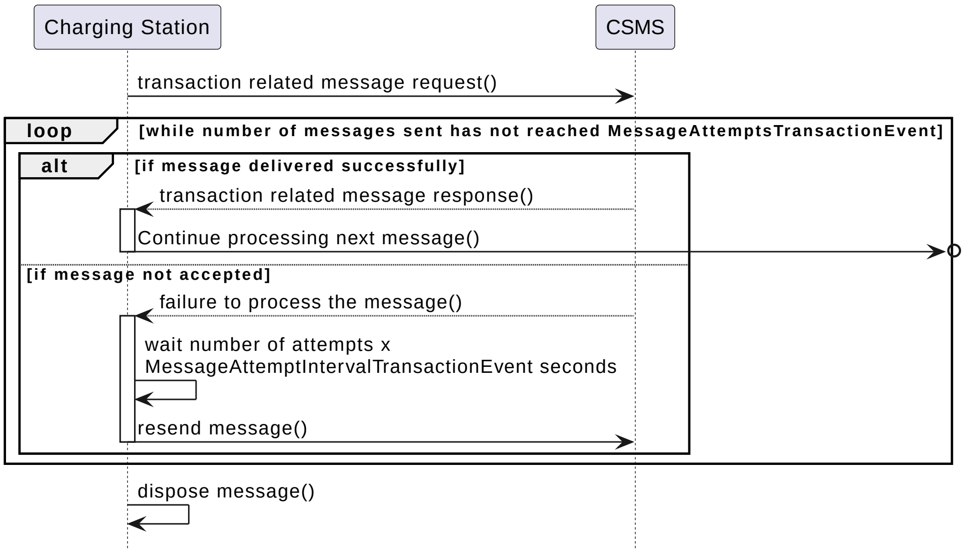

Charging Station
CSMS
transaction related message request()
loop
[while number of messages sent has not reached MessageAttemptsTransactionEvent]
alt
[if message delivered successfully]
transaction related message response()
Continue processing next message()
[if message not accepted]
failure to process the message()
wait number of attempts x
MessageAttemptIntervalTransactionEvent seconds
resend message()
dispose message()

Figure 72. Sequence Diagram: Transaction-related message not accepted by CSMS

E13 - Transaction-related message not accepted by CSMS - Requirements

Table 75. 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.

ID	Precondition	Requirement definition
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
3	**Objective(s)**	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 (eventType = Ended), 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	**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)**	**Successful postcondition:** The CSMS knows the status of the requested transaction. **Failure postcondition:** The CSMS does not know the status of the requested transaction.
7	**Error Handling**	n/a
8	**Remarks**	When the CSMS receives a GetTransactionStatusResponse with both fields (*ongoingIndicator* and *messagesInQueue*) 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).

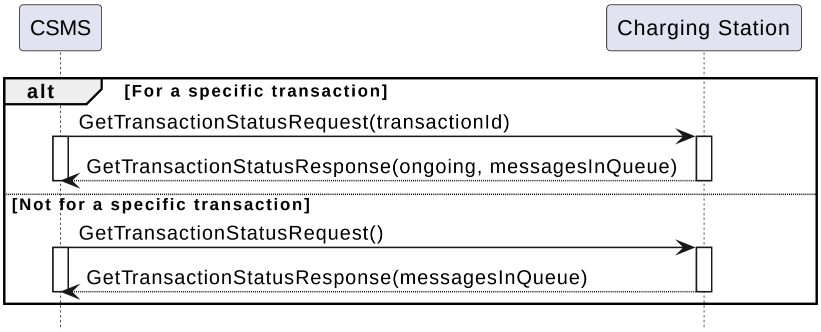

CSMS
Charging Station
alt
[For a specific transaction]
GetTransactionStatusRequest(transactionId)
GetTransactionStatusResponse(ongoing, messagesInQueue)
[Not for a specific transaction]
GetTransactionStatusRequest()
GetTransactionStatusResponse(messagesInQueue)

Figure 73. Sequence Diagram: Check transaction status

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 *ongoingIndicator* = false AND *messagesInQueue* = false.
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 *ongoingIndicator* = true.

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 *ongoingIndicator* = 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 *messagesInQueue* = 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 *messagesInQueue* = false.
E14.FR.06	The Charging Station receives a GetTransactionStatusRequest without a transactionId	The Charging Station's response SHALL NOT have *ongoingIndicator* 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 *messagesInQueue* = 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 *messagesInQueue* = false.

2.2. Interrupting and Stopping ISO 15118 Charging

E15 - End of charging process

No.	Type	Description
1	**Name**	End of charging process.
2	**ID**	E15
	Reference	ISO15118-1 H1 - End of charging process
3	**Objective(s)**	See ISO15118-1, use case Objective H1, page 44.
4	**Description**	See ISO15118-1, use case Description H1, page 44.
	Actors	EV, Charging Station, EV Driver
	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 eventType = Ended to the CSMS.
5	**Prerequisites**	See ISO15118-1, use case Prerequisites H1, page 44.
6	**Postcondition(s)**	The CSMS has received all relevant information about the transaction. See ISO15118-1, use case End Conditions H1, page 44.
7	**Error handling**	n/a

No.	Type	Description
8	Remark(s)	See ISO15118-1, use case Requirements H1, page 44 for the trigger. The scenario description and sequence diagram above are based on the Configuration Variable for stop transaction being configured as follows. `TxStopPoint`: ParkingBayOccupancy, EVConnected, Authorized, PowerPathClosed 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

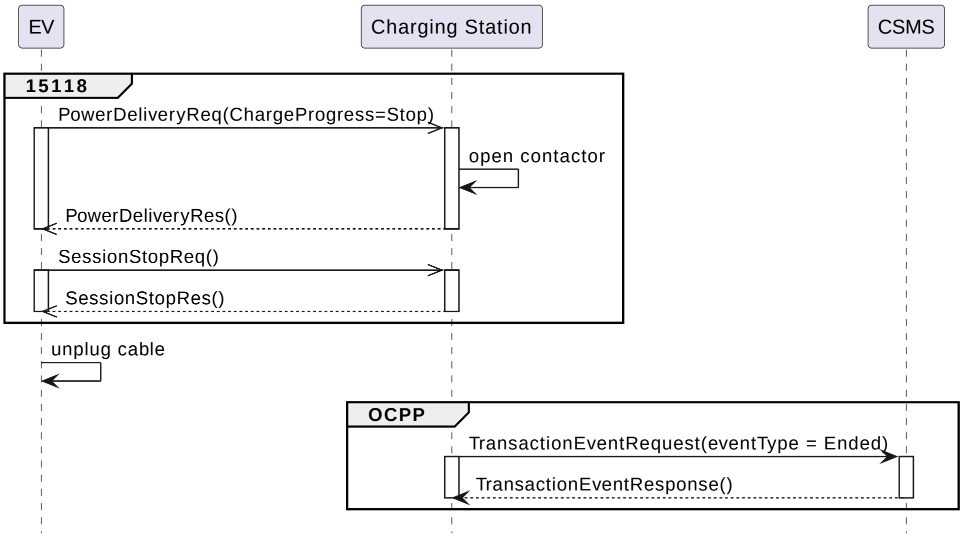


Figure 74. End of charging process

Source: ISO15118-1

E15 - End of charging process - Requirements

Table 76. 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. (Same as E08.FR.09)
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. (Same as E06.FR.12)
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. (Same as E06.FR.13)
E15.FR.04	When `TxStopPoint` contains "Authorized" or "PowerPathClosed" or "EnergyTransfer" AND Charging Station has not yet sent a TransactionEventRequest with *triggerReason* = `StopAuthorized` when it receives a ISO 15118 SessionStopReq(Terminate) message from the EV	Charging Station SHALL send a TransactionEventRequest message with eventType = Ended and *triggerReason* = `StopAuthorized` and *stoppedReason* = `StoppedByEV` to inform the CSMS that the charging transaction has been stopped (by the EV).
E15.FR.05	When `TxStopPoint` does not contain "Authorized" or "PowerPathClosed" or "EnergyTransfer" AND Charging Station has not yet sent a TransactionEventRequest with *triggerReason* = `StopAuthorized` when it receives a ISO 15118 SessionStopReq(Terminate) message from the EV	Charging Station SHALL send a TransactionEventRequest message with eventType = Updated and *triggerReason* = `StopAuthorized` to inform the CSMS that the authorization has ended.

2.3. Transactions with limits

E16 - Transactions with fixed cost, energy, SoC or time

New in OCPP 2.1

No.	Type	Description
1	**Name**	Transactions with fixed cost, energy, SoC or time
2	**ID**	E16
3	**Objective(s)**	EV Driver or CSMS specifies a limit in cost, energy, state of charge or time for transaction
4	**Description**	An EV Driver or CSMS specifies a limit in cost, energy, SoC or time. The limit can be updated during a transaction. A cost limit is most likely to be used for a prepaid charging session, or for a direct payment in which case the cost limit is set to the pre-authorized amount for the direct payment. An energy or time limit can also be provided by EV Driver or CSMS. Note that it is the energy transfer that is limited. It is not possible to limit the duration of a transaction, because the end of a transaction is determined by the configured TxStopPoint.
	Actors	Charging Station, CSMS, EV Driver
	Scenario description	*EV Driver specifies an energy limit* 1. EV Driver has input a maximum energy amount to charge via UI of Charging Station. 2. Charging Station includes *transactionInfo.transactionLimit.maxEnergy* with the specified maximum once in the next TransactionEventRequests for this transaction. 3. When the amount of energy imported by EV reaches *transactionInfo.transactionLimit.maxEnergy*, Charging Station stops the energy transfer and sends a TransactionEventRequest with *chargingState* = `SuspendedEVSE` and *triggerReason* = `EnergyLimitReached`. a. If the *transactionLimit* is increased when in this state, then Charging Station will resume the energy transfer.
	Scenario description #2	*CSMS specifies a cost limit* 1. CSMS has a cost limit for the transaction on this Charging Station. (This can be a fixed setting, a reservation amount for an adhoc payment, or the balance of a prepaid card as in use case C17 - Authorization with prepaid card). 2. In order to communicate this cost limit to the Charging Station, CSMS responds to a TransactionEventRequest with a TransactionEventResponse that has a *transactionLimit.maxCost* with the cost limit. This is only done once for every change in cost limit. 3. Charging Station will take note of this cost limit, and add the *transactionLimit.maxCost* element to *transactionInfo* of the next TransactionEventRequest message for this transaction as a confirmation towards CSMS of the received limit. 4. When the cost for energy transfer reaches *transactionInfo.transactionLimit.maxCost*, Charging Station stops the energy transfer and sends a TransactionEventRequest with *chargingState* = `SuspendedEVSE` and *triggerReason* = `CostLimitReached`. a. If the *transactionLimit* is increased when in this state, then Charging Station will resume the energy transfer. b. If Charging Station does not calculate the cost locally, then it stops the energy transfer as soon as *totalCost* in a TransactionEventResponse or CostUpdatedRequest reaches or is near the maximum cost.
5	**Prerequisite(s)**	Charging Station supports the limits maxEnergy and maxCost.
6	**Postcondition(s)**	Energy transfer in transaction is only allowed up to maximum cost, energy, state of charge or time limitation.
7	**Error handling**	Charging Station confirms a transaction limit from CSMS by returning the limit in the next TransactionEventRequest. If CSMS provides a transaction limit that Charging Station does not support, it will not be returned in the next TransactionEventRequest.

No.	Type	Description
8	**Remark(s)**	Only two scenarios are shown in the use case (energy limit and cost limit), but the same scenario is also valid for limits of time and state of charge that are supplied by either Charging Station or CSMS. It is possible to change the limit of a transaction even when energy transfer is already in progress, and it can be done more than once. If more than one limit is given, for example both a time and an energy limit, then whichever limit is reached first, determines the end of energy transfer. In order for a Charging Station to stop exactly on time when a cost limit is given, cost will have to be calculated locally at the Charging Station. If Charging Station does not support local cost calculation, and a cost limit is given, then it will use the value of the cost updates from CSMS (in TransactionEventResponse or CostUpdatedRequest) to determine when the cost limit is reached or exceeded. See section I Tariff and Cost for information on tariffs and cost calculation. A Charging Station reports the limits that it supports in configuration key "TxCtrlr.SupportedLimits".

NOTE Following diagrams only show parameters related to the use case. More parameters are required in these messages.
Two examples are shown. All limits can be set by either Charging Station or CSMS.

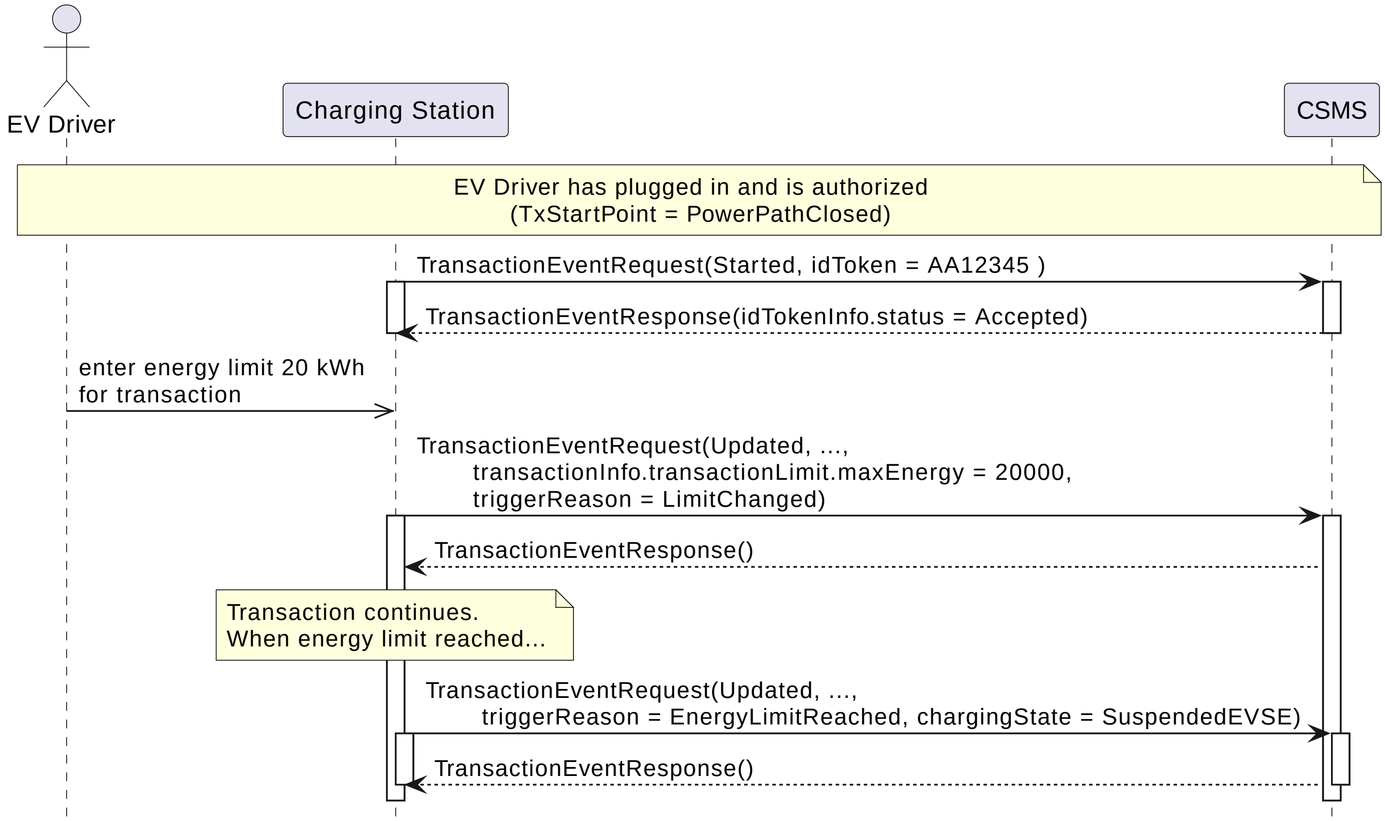

EV Driver
Charging Station
CSMS
EV Driver has plugged in and is authorized
(TxStartPoint = PowerPathClosed)
TransactionEventRequest(Started, idToken = AA12345)
TransactionEventResponse(idTokenInfo.status = Accepted)
enter energy limit 20 kWh
for transaction
TransactionEventRequest(Updated, ...,
transactionInfo.transactionLimit.maxEnergy = 20000,
triggerReason = LimitChanged)
TransactionEventResponse()
Transaction continues.
When energy limit reached...
TransactionEventRequest(Updated, ...,
triggerReason = EnergyLimitReached, chargingState = SuspendedEVSE)
TransactionEventResponse()

Figure 75. Sequence diagram for transaction with limited energy

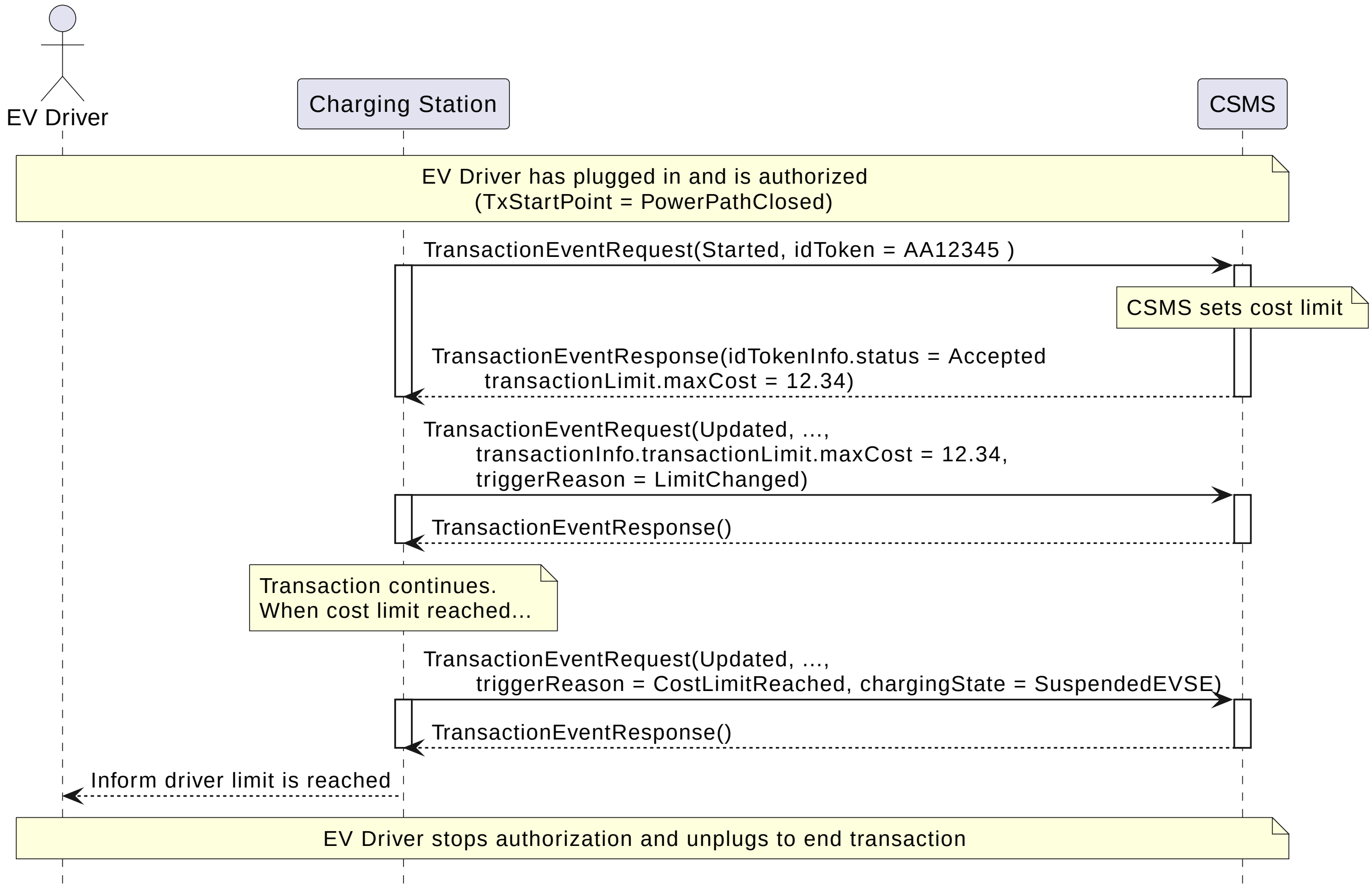

EV Driver
Charging Station
CSMS
EV Driver has plugged in and is authorized
(TxStartPoint = PowerPathClosed)
TransactionEventRequest(Started, idToken = AA12345)
CSMS sets cost limit
TransactionEventResponse(idTokenInfo.status = Accepted
transactionLimit.maxCost = 12.34)
TransactionEventRequest(Updated, ...,
transactionInfo.transactionLimit.maxCost = 12.34,
triggerReason = LimitChanged)
TransactionEventResponse()
Transaction continues.
When cost limit reached...
TransactionEventRequest(Updated, ...,
triggerReason = CostLimitReached, chargingState = SuspendedEVSE)
TransactionEventResponse()
Inform driver limit is reached
EV Driver stops authorization and unplugs to end transaction

Figure 76. Sequence diagram for transaction with limited cost

E16 - Transactions with fixed cost, energy, SoC or time - Requirements

ID	Precondition	Requirement definition	Note
E16.FR.01	When Charging Station sets or changes a limit to the transaction in cost, energy, SoC or time	Charging Station SHALL send a TransactionEventRequest with *triggerReason* = `LimitSet` and include this limit once in the field *transactionInfo.transactionLimit*.	This is usually a limit entered by EV Driver. Sending *transactionLimit* may be part of another *triggerReason* if multiple triggers apply, see E02.FR.17.
E16.FR.02	When CSMS sets or changes a limit to the transaction in cost, energy, SoC or time	CSMS SHALL include the field *transactionLimit* once in the first possible TransactionEventResponse message.	*transactionLimit* uses *maxCost* for cost limit, *maxEnergy* for energy limit, *maxTime* for time limit or *maxSoC* for SoC limit.
E16.FR.03	E16.FR.02 AND at least one of the limits in *transactionLimit* is supported by the Charging Station	Charging Station SHALL send a TransactionEventRequest with *triggerReason* = `LimitSet` and include the supported limit(s) once in the field *transactionInfo.transactionLimit*.	This confirms to CSMS that the limit is set. For example, a prepaid cost limit. See also E16.FR.13.
E16.FR.04	E16.FR.02	Charging Station SHALL NOT send a TransactionEventRequest with a *transactionInfo.transactionLimit* that exceeds a limit that was previously set by CSMS	For example, when CSMS has set a prepaid balance as a maximum cost, then CS must not allow EV Driver set the maximum cost to a higher limit.

E16.FR.05	During a transaction the *transactionLimit* is reached AND TxStopPoint does not contain "EnergyTransfer"	Charging Station SHALL suspend energy transfer AND send a TransactionEventRequest with *chargingState* = `SuspendedEVSE` and *triggerReason* `CostLimitReached`, `EnergyLimitReached`, `TimeLimitReached` or `SoCLimitReached`, depending on the reached limit.	The *triggerReason* for xxxLimitReached must be used, otherwise there is no way to know whether SuspendedEVSE is caused by smart charging or by a transaction limit.
E16.FR.06	During a transaction the *transactionLimit* is reached AND TxStopPoint contains "EnergyTransfer"	Charging Station SHALL end the transaction AND send a TransactionEventRequest with *eventType* = `Ended` and *triggerReason* `CostLimitReached`, `EnergyLimitReached`, `TimeLimitReached` or `SoCLimitReached`, depending on the reached limit.	The *triggerReason* for xxxLimitReached must be used, otherwise there is no way to know that end of transaction is caused by a transaction limit.
E16.FR.07	E16.FR.01 AND CSMS also wishes to set one or more of the same limits	CSMS MAY respond with TransactionEventResponse with an updated *transactionLimit* .	CSMS always has the last word and can overrule the limit from Charging Station.
E16.FR.08	E16.FR.07 AND The limit(s) reported by Charging Station are lower or equal to the limit(s) that CSMS requires	CSMS SHALL NOT respond with TransactionEventResponse with an updated *transactionLimit*.	The Charging Station limit is already below the limit that CSMS requires.
E16.FR.09		The timer for a time limit SHALL be relative to the start of the transaction (TransactionEventRequest(Started))	Purpose of a time limit is usually to limit the time that a station is kept occupied. Depending on TxStartPoint the start of a transaction can be some time before energy transfer.
E16.FR.10	When a *transactionLimit* is set at the time when the actual value already exceeds the limit	Charging Station SHALL immediately suspend energy transfer or stop the transaction according to E16.FR.05 or E16.FR.06.	Immediately act on exceeded limit. This also applies when a negative limit has been given.
E16.FR.11	When a *transactionLimit.maxCost* is active AND (TariffCostCtrlr[Tariff] = false OR when CSMS has not provided a (default) tariff plan)	CSMS SHALL provide cost updates to Charging Station in TransactionEventResponse or CostUpdatedRequest.	Central cost calculation is used.
E16.FR.12		CSMS SHALL NOT send a *transactionLimit* in a TransactionEventResponse that is not reported by the Charging Station in configuration key TxCtrlr.SupportedLimits.	
E16.FR.13	E16.FR.01 AND the limit is not supported by the Charging Station	Charging Station SHALL NOT return this element in the *transactionInfo.transactionLimit* field of the next TransactionEventRequest.	This tells CSMS that the limit is not supported.
E16.FR.14	When *chargingState* = `SuspendedEVSE` as a result of reaching a *transactionLimit* (E16.FR.06) AND The *transactionLimit* is increased (either by EV Driver or CSMS)	Charging Station SHALL resume energy transfer.	
E16.FR.15	E16.FR.11	Charging Station SHALL use the cost update from CSMS to determine whether cost limit has been reached or exceeded.	
E16.FR.16	When a *transactionLimit.maxCost* is active AND (TariffCostCtrlr[Tariff] = true AND CSMS has provided a (default) tariff plan)	Charging Station SHALL NOT use the cost updates from CSMS in TransactionEventResponse or CostUpdatedRequest to determine the cost, but use local cost calculation via the tariff plan.	Local cost calculation is used.

E16.FR.17	If CSMS or Charging Station wish to remove a certain transaction limit	CSMS or Charging Station SHALL set the value of this limit to a high enough value, such that it is no longer relevant as a limit.	Once a limit is set in a transaction, it cannot be unset, only changed.
E16.FR.18	When CSMS sends a TransactionEventResponse with a *transactionLimit* in response to a TransactionEventRequest message with *offline* = true	CSMS SHOULD NOT expect a confirmation from Charging Station for the *transactionLimit* in TransactionEventRequest messages that have *offline* = true.	The queued messages with *offline* = true are historical messages from the offline period and not a response to setting the *transactionLimit*. See also E16.FR.03.
E16.FR.19	When the granularity of the measurements on Charging Station of *transactionLimit* values is such that it may not be possible to stop exactly on the limit	Charging Station SHOULD try to avoid exceeding the *transactionLimit* values.	For example, if an energy meter only reports in 1000 Wh increments and the *maxEnergy* limit is set to 2300 Wh, then Charging Station should stop at 2000 Wh.
E16.FR.20	When Charging Station receives TransactionEventResponse with a *transactionLimit* that is does not support	Charging Station MAY report this as a NotifyEventRequest message with *trigger* = `Alerting` and *eventNotificationType* = `HardWiredNotification` for *component* = "TxCtrlr", *variable* = "SupportedLimits" and *actualValue* set to the not-supported limit: "maxEnergy", "maxCost", "maxTime" or "maxSoC".	See also E16.FR.12

2.4. Resuming transactions

E17 - Resuming transaction after forced reboot

New in OCPP 2.1

No.	Type	Description
1	**Name**	Resuming transaction after forced reboot
2	**ID**	E17
3	**Objective(s)**	To show how a transaction can be resumed after it has been interrupted as result of a forced reboot of the Charging Station. This allows a transaction to survive interruptions, such as power outages or critical software faults.
4	**Description**	A transaction is running at the Charging Station. The Charging Station gets shut down unexpectedly, e.g. due to a software fault, watchdog event, maintenance mode activation or power loss at the Charging Station. If the interruption lasts no longer than the configured `TxResumptionTimeout`, then the Charging Station will resume the transaction. If the interruption exceeds the timeout, then the transaction will end. The configuration variable `TxAllowEnergyTransferResumption` can be set to false if for a resumed transaction the automatic resumption of energy transfer to the vehicle is not allowed.
	Actors	Charging Station, CSMS
	Scenario description #1	*Resuming transaction before `TxResumptionTimeout` seconds* 1. Configuration variable `TxResumptionTimeout` has been set to a value > 0. 2. An ongoing transaction is interrupted, for example as a result of power loss. 3. Charging Station reboots when power is restored. 4. If the interruption occurred less (or equal) than `TxResumptionTimeout` seconds ago, then Charging Station resumes the transaction in the charging state from before the interruption. a. If `TxAllowEnergyTransferResumption` = false then energy transfer is not allowed anymore on the resumed transaction. b. Charging State sends TransactionEventRequest with *eventType* = `Updated` and *triggerReason* = `TxResumed`.

No.	Type	Description
	Scenario description #2	*Ending transaction after* `TxResumptionTimeout` *seconds* 1. Configuration variable `TxResumptionTimeout` has been set to a value > 0. 2. An ongoing transaction is interrupted, for example as a result of power loss. 3. Charging Station reboots when power is restored. 4. If the interruption occurred more than `TxResumptionTimeout` seconds ago, then Charging Station ends the transaction. a. Charging Station sends TransactionEventRequest with *eventType* = `Ended` and *triggerReason* = `AbnormalCondition` and *stoppedReason* = `PowerLoss` (or `Reboot` if Charging Station rebooted because of a local issue).
5	**Prerequisite(s)**	`TxResumptionTimeout` has been set.
6	**Postcondition(s)**	Transactions survive forced reboot. **Failure postcondition:** - The Charging Station was not able to resume the transaction, - The EV was not able to resume the energy transfer.
7	**Error handling**	
8	**Remark(s)**	The EVSE of the Charging Station will have to resume the charging session towards the EV. Depending on the type of connector and protocol (Mode 3, ISO 15118, CHAdeMO, etc.) this will involve different actions, which are not detailed in this specification. Charging profiles of type `TxProfile` are not required to be stored in persistent memory. If such a charging profile is applicable to the resumed transaction(s), then CSMS will have to send the charging profile again. `TxAllowEnergyTransferResumption` = false can be used to protect the user from a scenario where e.g. the plug has been removed and inserted into another vehicle during the interruption, therefore enabling another EV driver to charge based on a different EV driver's transaction and authorization.

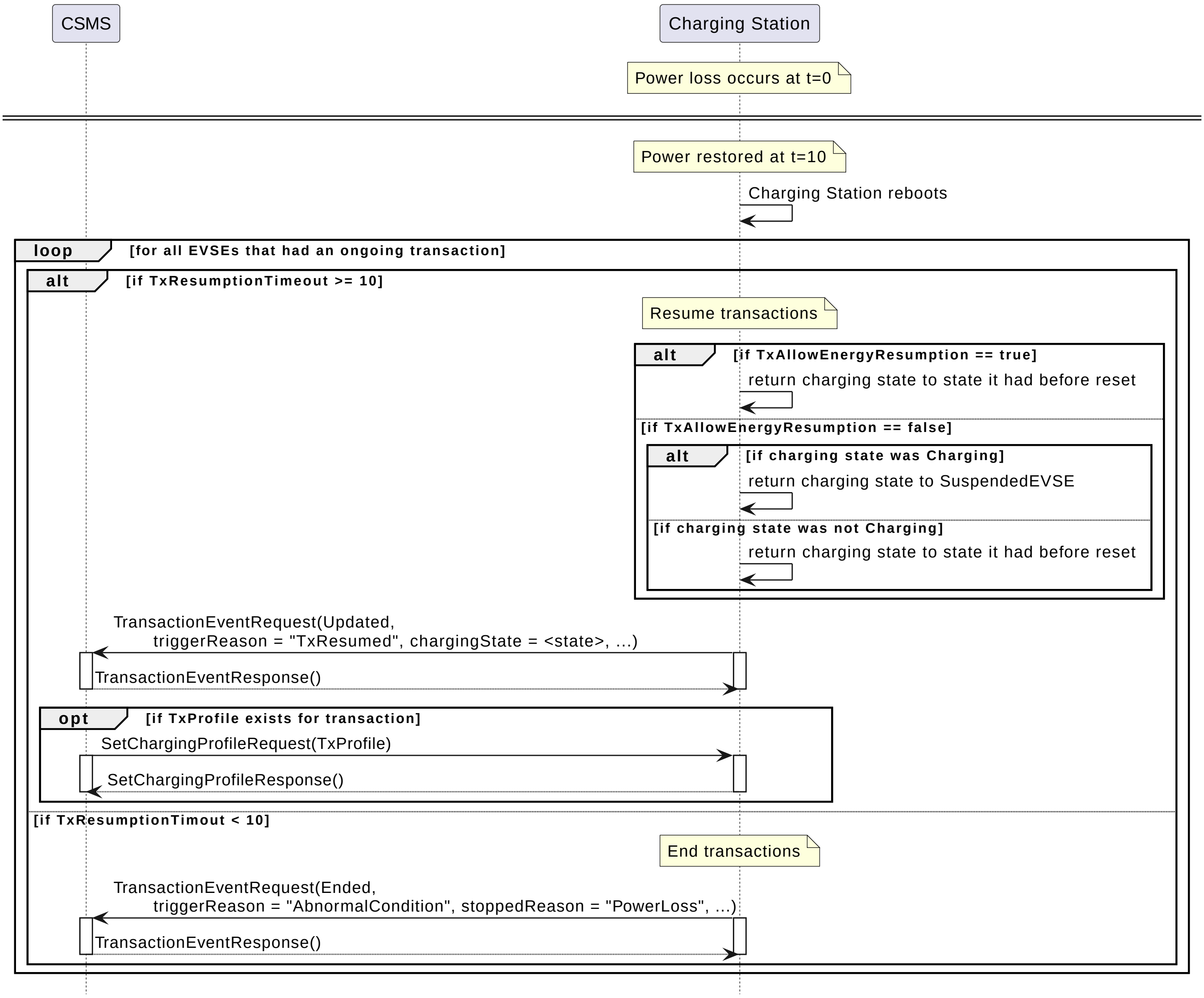

CSMS
Charging Station
Power loss occurs at t=0
Power restored at t=10
Charging Station reboots
loop [for all EVSEs that had an ongoing transaction]
alt [if TxResumptionTimeout >= 10]
Resume transactions
alt [if TxAllowEnergyResumption == true]
return charging state to state it had before reset
[if TxAllowEnergyResumption == false]
alt [if charging state was Charging]
return charging state to SuspendedEVSE
[if charging state was not Charging]
return charging state to state it had before reset
TransactionEventRequest(Updated, triggerReason = "TxResumed", chargingState = <state>, ...)
TransactionEventResponse()
opt [if TxProfile exists for transaction]
SetChargingProfileRequest(TxProfile)
SetChargingProfileResponse()
[if TxResumptionTimout < 10]
End transactions
TransactionEventRequest(Ended, triggerReason = "AbnormalCondition", stoppedReason = "PowerLoss", ...)
TransactionEventResponse()

Figure 77. Sequence diagram for resuming an interrupted transaction

E17 - Resuming transaction after interruption - Requirements

ID	Precondition	Requirement definition	Note
		Generic	
E17.FR.01	If `TxResumptionTimeout` > 0	Charging Station SHALL store transaction state in persistent memory	This is needed in order to resume transactions after a power loss.
E17.FR.02	E17.FR.01	The timer for `TxResumptionTimeout` starts from the moment that Charging Station detected or had recorded the unexpected shutdown	It is up to the implementer how time of shutdown is determined after restarting.
		Within timeout	
E17.FR.10	When Charging Station reboots after interruption AND Interruption was not longer than `TxResumptionTimeout` seconds	Charging Station SHALL resume the transactions that were ongoing before the interruption.	
E17.FR.11	E17.FR.10 AND `TxAllowEnergyTransferResumption` = true AND the EV and EVSE are not disconnected	Charging Station SHALL resume such a transaction in the charging state it had before the interruption.	

E17.FR.12	E17.FR.10 AND `TxAllowEnergyTransferResumption` = false AND Charging state was `Charging` AND the EV and EVSE are not disconnected	Charging Station SHALL resume such a transaction in the charging state `SuspendedEVSE` or `EVConnected`.	
E17.FR.13	E17.FR.10 AND `TxAllowEnergyTransferResumption` = false AND Charging state was **not** (`Charging` or `SuspendedEV`) AND the EV and EVSE are not disconnected	Charging Station SHALL resume such a transaction in the charging state it had before the interruption.	
E17.FR.14		For each resumed transaction Charging Station SHALL send a TransactionEventRequest with *eventType* = `Updated` and *triggerReason* = `TxResumed` and its *chargingState* as well as other relevant parameters.	
E17.FR.15	E17.FR.14 AND CSMS has a charging profile of *chargingProfilePurpose* = `TxProfile` for transactions reported with *triggerReason* = `TxResumed` AND SmartChargingCtrlr.ChargingProfilePersistence for *instance* = "TxProfile" = false or absent	CSMS SHALL send the SetChargingProfileRequest for *chargingProfilePurpose* = `TxProfile` for these transactions again.	The charging profiles are identical to those sent at start of transaction, since start time of the resumed transaction has not changed.
E17.FR.16	E17.FR.10 AND the EV and EVSE are disconnected AND the TxStopPoint conditions are not met	Charging Station SHALL resume such a transaction in the charging state `Idle`.	
E17.FR.17	E17.FR.10 AND the EV and EVSE are disconnected AND the TxStopPoint conditions are met	The transaction will end normally, as described by E06 - Stop Transaction options	
E17.FR.18	E17.FR.11 AND Before interruption the charging state was `Charging` AND The EV and EVSE are not disconnected	The Charging Station SHALL delay resuming energy transfer for a random amount of seconds, with a maximum of the value configured at `EnergyTransferResumptionRandomRange`.	If after a power outage all Charging Stations in an area start resuming energy transfer at the exact same time, this may result in issues with the energy grid.
After timeout			
E17.FR.20	When Charging Station reboots after interruption AND Interruption was longer than `TxResumptionTimeout` seconds	Charging Station SHALL end the transactions that were ongoing before the interruption.	
E17.FR.21	E17.FR.20 AND Interruption was caused by a power loss	Charging Station SHALL send a TransactionEventRequest with *eventType* = `Ended` and *triggerReason* = `AbnormalCondition` and *stoppedReason* = `PowerLoss`.	
E17.FR.22	E17.FR.20 AND Interruption was caused by Charging Station (e.g. software fault or watch dog timer)	Charging Station SHALL send a TransactionEventRequest with *eventType* = `Ended` and *triggerReason* = `AbnormalCondition` and *stoppedReason* = `Reboot`.	

F. Remote Control

Chapter 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.

Chapter 2. Use cases & Requirements

2.1. Remote Transaction Control

F01 - Remote Start Transaction - Cable Plugin First

No.	Type	Description
1	**Name**	Remote Start Transaction - Cable Plugin First
2	**ID**	F01
3	**Objective(s)**	**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	**1.** The EV Driver plugs in the cable at the Charging Station. **2.** The Charging Station sends a NotifyEventRequest with component.name *Connector*, variable.name *AvailabilityState* and actualValue *Occupied* to the CSMS to inform it about a Connector that became *Occupied* **3.** The CSMS responds with a NotifyEventResponse, confirming that the NotifyEventRequest was received. **4.** The Charging Station sends a TransactionEventRequest (eventType = Started) 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 *transactionId* 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 (eventType = Updated, chargingState = Charging) 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 *true*, the Charging Station will only offer energy when it successfully authorized the IdToken, using Local Authorization List, Authorization Cache and/or an AuthorizeRequest.
7	**Error handling**	n/a
8	**Remark(s)**	An external trigger can be e.g. a Charging Station Operator or an EV Driver app. The RequestStartTransactionResponse contains a status which indicates whether the Charging Station has accepted the request and will attempt to start a transaction. 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. The scenario description and sequence diagram above are based on the Configuration Variable for start transaction being configured as follows: `TxStartPoint`: 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 send. For more details see the use cases: E01 - Start Transaction options.

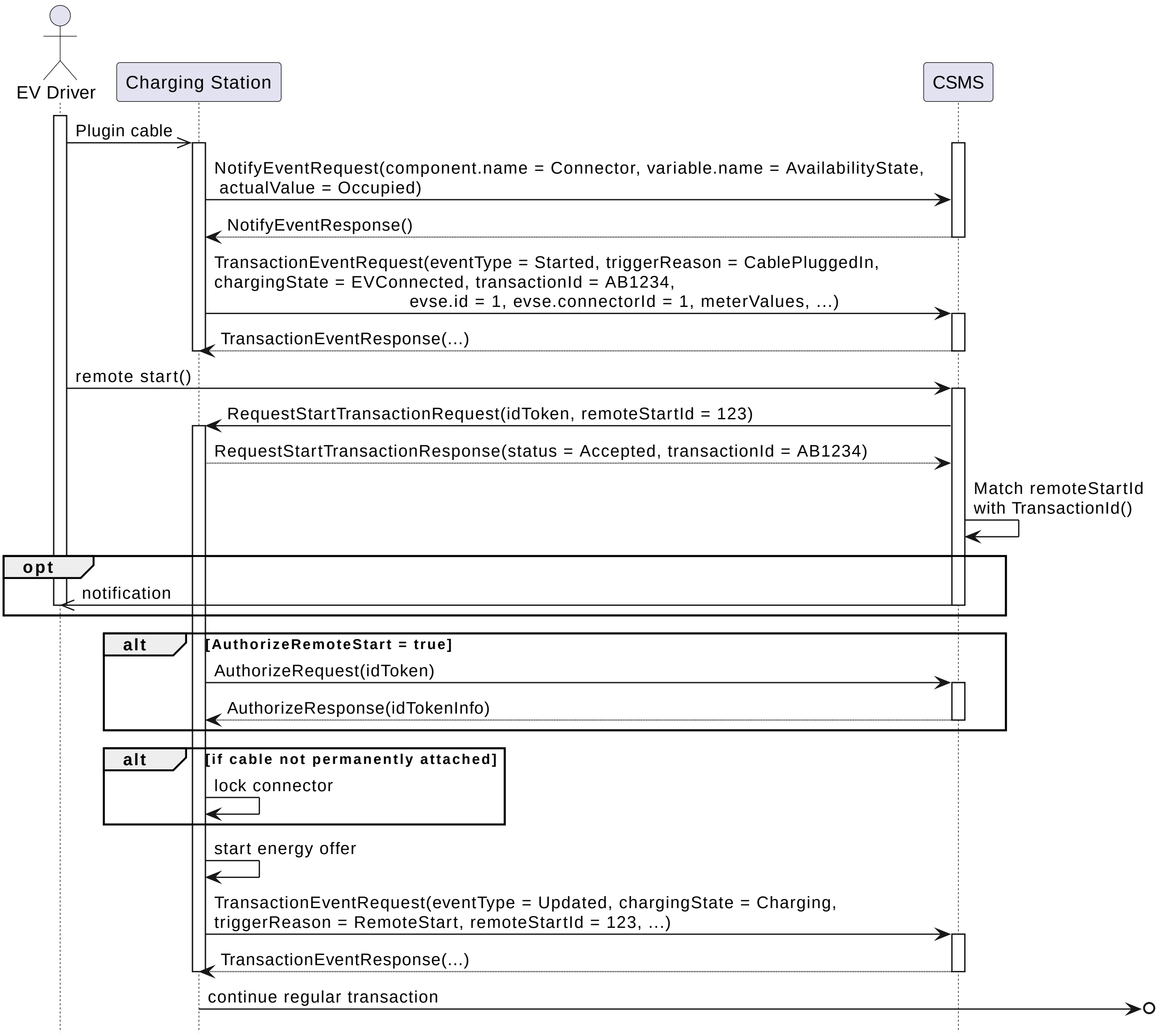

EV Driver
Charging Station
CSMS
Plugin cable
NotifyEventRequest(component.name = Connector, variable.name = AvailabilityState, actualValue = Occupied)
NotifyEventResponse()
TransactionEventRequest(eventType = Started, triggerReason = CablePluggedIn, chargingState = EVConnected, transactionId = AB1234, evse.id = 1, evse.connectorId = 1, meterValues, ...)
TransactionEventResponse(...)
remote start()
RequestStartTransactionRequest(idToken, remoteStartId = 123)
RequestStartTransactionResponse(status = Accepted, transactionId = AB1234)
Match remoteStartId with TransactionId()
opt
notification
alt
[AuthorizeRemoteStart = true]
AuthorizeRequest(idToken)
AuthorizeResponse(idTokenInfo)
alt
[if cable not permanently attached]
lock connector
start energy offer
TransactionEventRequest(eventType = Updated, chargingState = Charging, triggerReason = RemoteStart, remoteStartId = 123, ...)
TransactionEventResponse(...)
continue regular transaction

Figure 78. Sequence Diagram: Remote Start Transaction - Cable Plugged in First

F01 - Remote Start Transaction - Cable Plugin First - Requirements

Table 77. 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.

ID	Precondition	Requirement definition	Note
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 evseId if the transaction is to be started on a specific EVSE.	When no evseId is provided, the Charging Station is in control of the EVSE selection.
F01.FR.07	If the RequestStartTransactionRequest does not contain an evseId.	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.
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 *transactionId* in the RequestStartTransactionResponse.	
F01.FR.14	When configured to send meter data in the TransactionEventRequest (eventType = Started), See: Meter Values - Configuration	The Charging Station SHALL add the configured measurands to the optional meterValue field in the TransactionEventRequest(eventType = Started) sent to the CSMS to provide more details during the transaction.	
F01.FR.15	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.	Same as J02.FR.11

ID	Precondition	Requirement definition	Note
F01.FR.16	F01.FR.15 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*.	Same as J02.FR.14
F01.FR.17	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.	Same as E01.FR.15
F01.FR.18	After a transaction has been started	The Charging Station MAY send additional TransactionEventRequest(eventType = Updated) messages during the transaction when a trigger event occurs.	
F01.FR.19	When a RequestStartTransactionRequest is received.	The next TransactionEventRequest SHALL contain *triggerReason*: RemoteStart.	
F01.FR.20	If the RequestStartTransactionRequest does not contain an *evseId* AND the Charging Station is capable of selecting an EVSE	The Charging Station SHALL select an EVSE to be used as a value for *evseId* for the operation	See also F01.FR.07 if Charging Station does not support starting at an arbitrary EVSE.
F01.FR.21	When the *evseId* for RequestStartTransactionRequest is `Reserved` for an *idToken* that differs from *idToken* in the request AND has no reservation for a *groupIdToken*	The Charging Station SHALL respond with RequestStartTransactionResponse with *status* = `Rejected`.	
F01.FR.22	When the *evseId* for RequestStartTransactionRequest is `Reserved` for an *idToken* that differs from *idToken* in the request AND is `Reserved` for a *groupIdToken* that differs from *groupIdToken* in the request	The Charging Station SHALL respond with RequestStartTransactionResponse with *status* = `Rejected`.	EV is not allowed to use station if neither *idToken* nor *idGroupToken* match the reservation.
F01.FR.23	When the *evse* for RequestStartTransactionRequest is `Unavailable` or `Faulted`	The Charging Station SHALL respond with RequestStartTransactionResponse with *status* = `Rejected`.	
F01.FR.24	When the *evseId* for RequestStartTransactionRequest is `Occupied` AND this *evseId* 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 RequestStartTransactionRequest
F01.FR.25	F01.FR.13	The Charging Station SHALL put the *remoteStartId* in the next TransactionEventRequest it sends for the associated transaction.	
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 *reasonCode* = "InvalidProfile" or "InvalidSchedule".	The device model variable SmartChargingCtrlr.Enabled tells CSMS whether smart charging is supported.

F02 - Remote Start Transaction - Remote Start First

No.	Type	Description
1	**Name**	Remote Start Transaction - Remote Start first
2	**ID**	F02
	Parent use case	F01 - Remote Start Transaction - Cable Plugin First

No.	Type	Description
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
	Scenario description	**1.** An External Trigger triggers the remote start. **2.** The CSMS sends RequestStartTransactionRequest to the Charging Station. **3.** The Charging Station responds with RequestStartTransactionResponse to the CSMS. **4.** The EV Driver is authorized by the CSMS, dependent on the Configuration Variable settings. **5.** The Charging Station sends a TransactionEventRequest (eventType = Started) notifying the CSMS about a transaction that has started **6.** The cable is plugged in. **6a.** Charging Station sends a NotifyEventRequest with component.name *Connector*, variable.name *AvailabilityState* and actualValue *Occupied* **6b.** CSMS sends a NotifyEventResponse to the Charging Station **7.** The energy offer is started. **8.** The Charging Station sends a TransactionEventRequest (eventType = Updated, chargingState = Charging) message to inform the CSMS that the charging has started. **9.** The CSMS sends TransactionEventResponse to the Charging Station
5	**Prerequisite(s)**	Charging Cable not plugged in. Remote start first. Enable mobile apps to control charging transactions via the CSMS.
6	**Postcondition(s)**	**Successful postcondition:** The transaction for which a start was request has started and the EV is charging. **Failure postcondition:** The transaction for which a start was request did not start or the EV is not charging.
7	**Error handling**	n/a
8	**Remark(s)**	An external trigger can be e.g. a Charging Station Operator or an EV Driver app. It is advised not to start transactions remotely without evseId 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. 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. The scenario description and sequence diagram above are based on the Configuration Variable for start transaction being configured as follows: `TxStartPoint`: either EVConnected or Authorized 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.

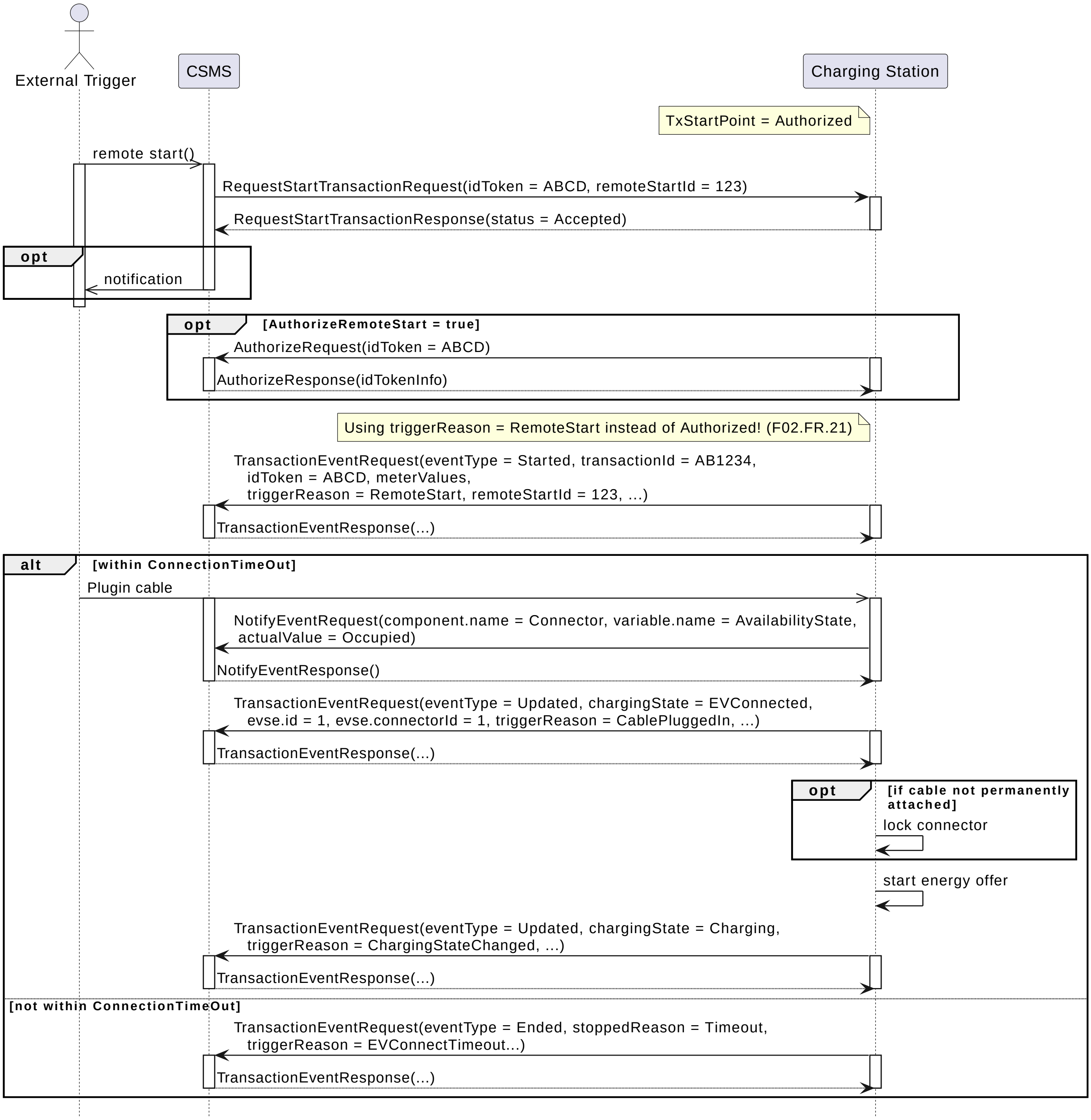

External Trigger
CSMS
Charging Station
TxStartPoint = Authorized
remote start()
RequestStartTransactionRequest(idToken = ABCD, remoteStartId = 123)
RequestStartTransactionResponse(status = Accepted)
opt
notification
opt
[AuthorizeRemoteStart = true]
AuthorizeRequest(idToken = ABCD)
AuthorizeResponse(idTokenInfo)
Using triggerReason = RemoteStart instead of Authorized! (F02.FR.21)
TransactionEventRequest(eventType = Started, transactionId = AB1234, idToken = ABCD, meterValues, triggerReason = RemoteStart, remoteStartId = 123, ...)
TransactionEventResponse(...)
alt
[within ConnectionTimeOut]
Plugin cable
NotifyEventRequest(component.name = Connector, variable.name = AvailabilityState, actualValue = Occupied)
NotifyEventResponse()
TransactionEventRequest(eventType = Updated, chargingState = EVConnected, evse.id = 1, evse.connectorId = 1, triggerReason = CablePluggedIn, ...)
TransactionEventResponse(...)
opt
[if cable not permanently attached]
lock connector
start energy offer
TransactionEventRequest(eventType = Updated, chargingState = Charging, triggerReason = ChargingStateChanged, ...)
TransactionEventResponse(...)
[not within ConnectionTimeOut]
TransactionEventRequest(eventType = Ended, stoppedReason = Timeout, triggerReason = EVConnectTimeout...)
TransactionEventResponse(...)

Figure 79. Sequence Diagram: Remote Start Transaction - Remote Start First with TxStartPoint=Authorized

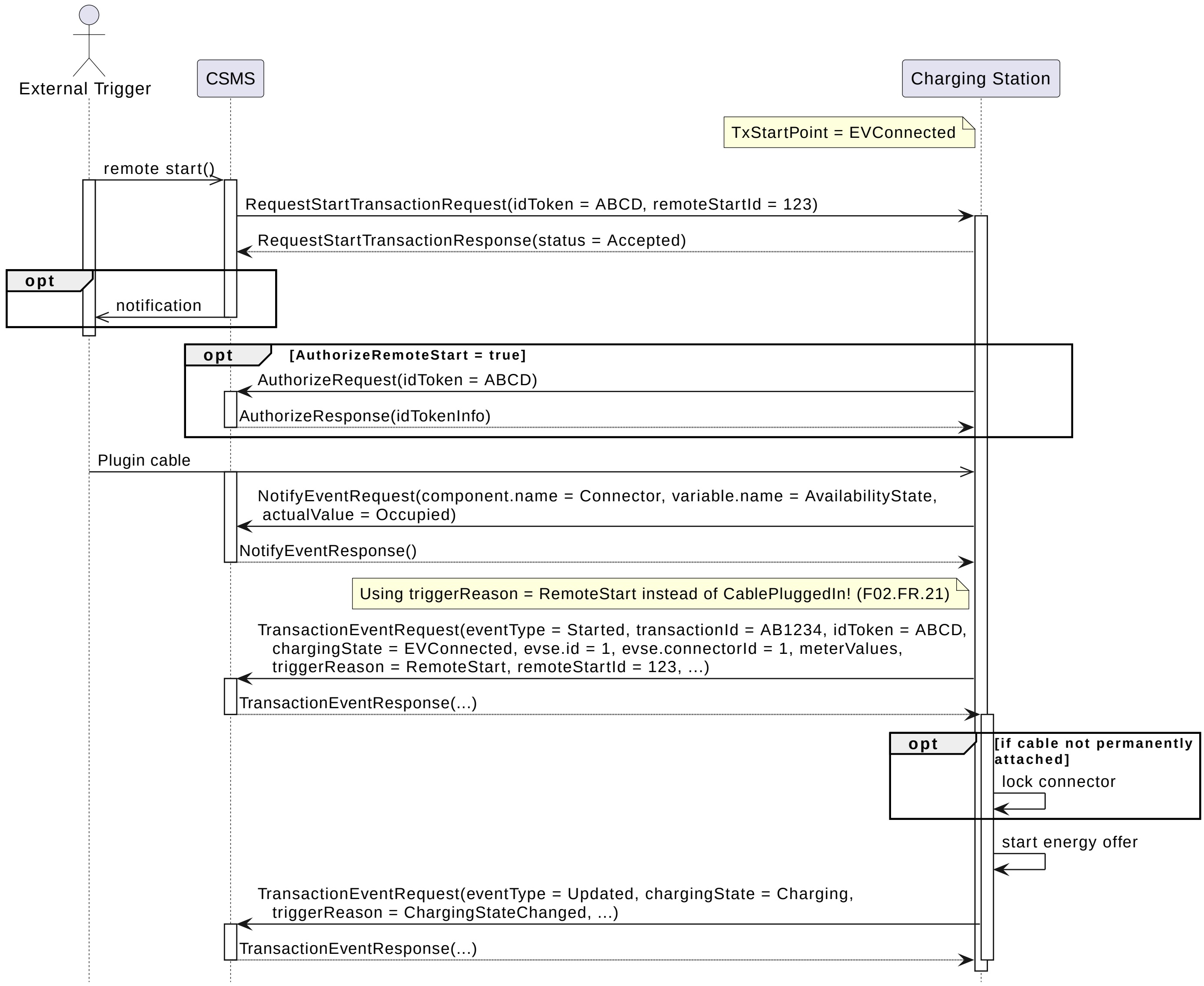

External Trigger
CSMS
Charging Station
TxStartPoint = EVConnected
remote start()
RequestStartTransactionRequest(idToken = ABCD, remoteStartId = 123)
RequestStartTransactionResponse(status = Accepted)
opt
notification
opt
[AuthorizeRemoteStart = true]
AuthorizeRequest(idToken = ABCD)
AuthorizeResponse(idTokenInfo)
Plugin cable
NotifyEventRequest(component.name = Connector, variable.name = AvailabilityState, actualValue = Occupied)
NotifyEventResponse()
Using triggerReason = RemoteStart instead of CablePluggedIn! (F02.FR.21)
TransactionEventRequest(eventType = Started, transactionId = AB1234, idToken = ABCD, chargingState = EVConnected, evse.id = 1, evse.connectorId = 1, meterValues, triggerReason = RemoteStart, remoteStartId = 123, ...)
TransactionEventResponse(...)
opt
[if cable not permanently attached]
lock connector
start energy offer
TransactionEventRequest(eventType = Updated, chargingState = Charging, triggerReason = ChargingStateChanged, ...)
TransactionEventResponse(...)

Figure 80. Sequence Diagram: Remote Start Transaction - Remote Start First with TxStartPoint=EVConnected

F02 - Remote Start Transaction - Remote Start First - Requirements

Table 78. 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 *remoteStartId* in the first TransactionEventRequest it sends for this new transaction.	
F02.FR.02	When configured to send meter data in the TransactionEventRequest (eventType = Started), See: Meter Values - Configuration	The Charging Station SHALL add the configured measurands to the optional meterValue field in the TransactionEventRequest(eventType = Started) sent to the CSMS to provide more details during the transaction.	Same as F01.FR.14
F02.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 meterValue field in the TransactionEventRequest(eventType = Updated) sent to the CSMS to provide more details during the transaction.	Same as J02.FR.11
F02.FR.04	F02.FR.03 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*.	Same as J02.FR.14
F02.FR.05	When the IdToken information is known.	The next TransactionEventRequest SHALL contain IdTokenType information.	Same as E03.FR.01

ID	Precondition	Requirement definition	Note
F02.FR.06	This transaction ends a reservation for the specific IdToken.	The next TransactionEventRequest SHALL contain the reservationId.	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 (eventType = Ended, stoppedReason = Timeout, *triggerReason* = `EVConnectionTimeout`) to the CSMS.	Otherwise the transaction would not be ended in case the TxStopPoint does not contain `Authorized`. Same as E03.FR.05
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 (*triggerReason* = `EVConnectionTimeout`) to the CSMS.	Transaction will be ended normally when driver leaves the parking bay. Same as E03.FR.15
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 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. A transaction is only started after authorization was obtained. Same as F01.FR.01
F02.FR.10	If the value of `AuthorizeRemoteStart` = false AND Charging Station receives a RequestStartTransactionRequest	The Charging Station SHALL start a transaction for the IdToken given in RequestStartTransactionRequest message without checking authorization.	Note that after the transaction has been started, the Charging Station will send a TransactionEventRequest with the idToken to the CSMS, and the CSMS will check the authorization status of the IdToken when processing this TransactionEventRequest. Same as F01.FR.02
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 IdToken when processing this TransactionEventRequest.	Same as F01.FR.03
F02.FR.12		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.	Same as F01.FR.04
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 evseId if the transaction is to be started on a specific EVSE.	When no evseId is provided, the Charging Station is in control of the EVSE selection. Same as F01.FR.06
F02.FR.15	If the RequestStartTransactionRequest does not contain an evseId.	The Charging Station MAY reject the RequestStartTransactionRequest.	Same as F01.FR.07
F02.FR.16		The CSMS MAY include a ChargingProfile in the RequestStartTransactionRequest.	Same as F01.FR.08
F02.FR.17	F02.FR.16	The purpose of this ChargingProfile SHALL be set to TxProfile.	Same as F01.FR.09

ID	Precondition	Requirement definition	Note
F02.FR.18	F02.FR.16	The Charging Station SHALL use this ChargingProfile for the transaction that is started by this RequestStartTransaction.	Same as F01.FR.10
F02.FR.19	F02.FR.16	The transactionId in the ChargingProfile SHALL NOT be set.	Same as F01.FR.11
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 SmartChargingCtrlr.Enabled tells CSMS whether smart charging is supported. Same as F01.FR.12
F02.FR.21	When a RequestStartTransactionRequest is received.	The next TransactionEventRequest SHALL contain *triggerReason*: `RemoteStart` and the *remoteStartId* from the RequestStartTransactionRequest.	This is to notify CSMS that this is the result of RequestStartTransaction. Note, that if TxStartPoint=`EVConnected` the transaction will be started upon cable connection, but the *triggerReason* = `RemoteStart` must still be sent. The connection event is reported by the fact that *chargingState* = `EVConnected`.
F02.FR.22	If the RequestStartTransactionRequest does not contain an *evseId* AND the Charging Station is capable of selecting an EVSE	The Charging Station SHALL select an EVSE to be used as a value for *evseId* for the operation	See also F02.FR.15 if Charging Station does not support starting at an arbitrary EVSE. Same as F01.FR.20
F02.FR.23	When the *evseId* for RequestStartTransactionRequest is `Reserved` for an *idToken* that differs from *idToken* in the request AND has no reservation for a *groupIdToken*	The Charging Station SHALL respond with RequestStartTransactionResponse with *status* = `Rejected`.	Same as F01.FR.21
F02.FR.24	When the *evseId* for RequestStartTransactionRequest is `Reserved` for an *idToken* that differs from *idToken* in the request AND is `Reserved` for a *groupIdToken* that differs from *groupIdToken* in the request	The Charging Station SHALL respond with RequestStartTransactionResponse with *status* = `Rejected`.	EV is not allowed to use station if neither *idToken* nor *idGroupToken* match the reservation. Same as F01.FR.22
F02.FR.25	When the *evseId* for RequestStartTransactionRequest is `Unavailable` or `Faulted`	The Charging Station SHALL respond with RequestStartTransactionResponse with *status* = `Rejected`.	Same as F01.FR.23
F02.FR.26	When the *evseId* for RequestStartTransactionRequest is `Occupied` AND this *evseId* 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 RequestStartTransactionRequest. Same as F01.FR.24
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 *reasonCode* = "InvalidProfile" or "InvalidSchedule".	The device model variable SmartChargingCtrlr.Enabled tells CSMS whether smart charging is supported. Same as F01.FR.26

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

No.	Type	Description
1	**Name**	Remote Stop Transaction
2	**ID**	F03
3	**Objective(s)**	**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.
	Actors	Charging Station, CSMS, CSO, EV Driver
	Scenario description	**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 (eventType = Updated) and, if applicable, unlocks the Connector. **5.** After the EV Driver unplugs the cable, the Charging Station sendsNotifyEventRequest with component.name *Connector*, variable.name *AvailabilityState* and actualValue *Available* **6.** The Charging Station ends the transaction and sends a TransactionEventRequest (eventType = *Ended*, stoppedReason = *Remote*) message to the CSMS.
5	**Prerequisite(s)**	A transaction is ongoing.
6	**Postcondition(s)**	**Successful postcondition:** The transaction for which a stop was request has ended. **Failure postcondition:** The transaction for which a stop was requested is still ongoing.
7	**Remark(s)**	This remote request to stop a transaction is equal to a local action to stop a transaction. The scenario description and sequence diagram above are based on the Configuration Variable for stop transaction being configured as follows. `TxStopPoint`: ParkingBayOccupancy, EVConnected 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

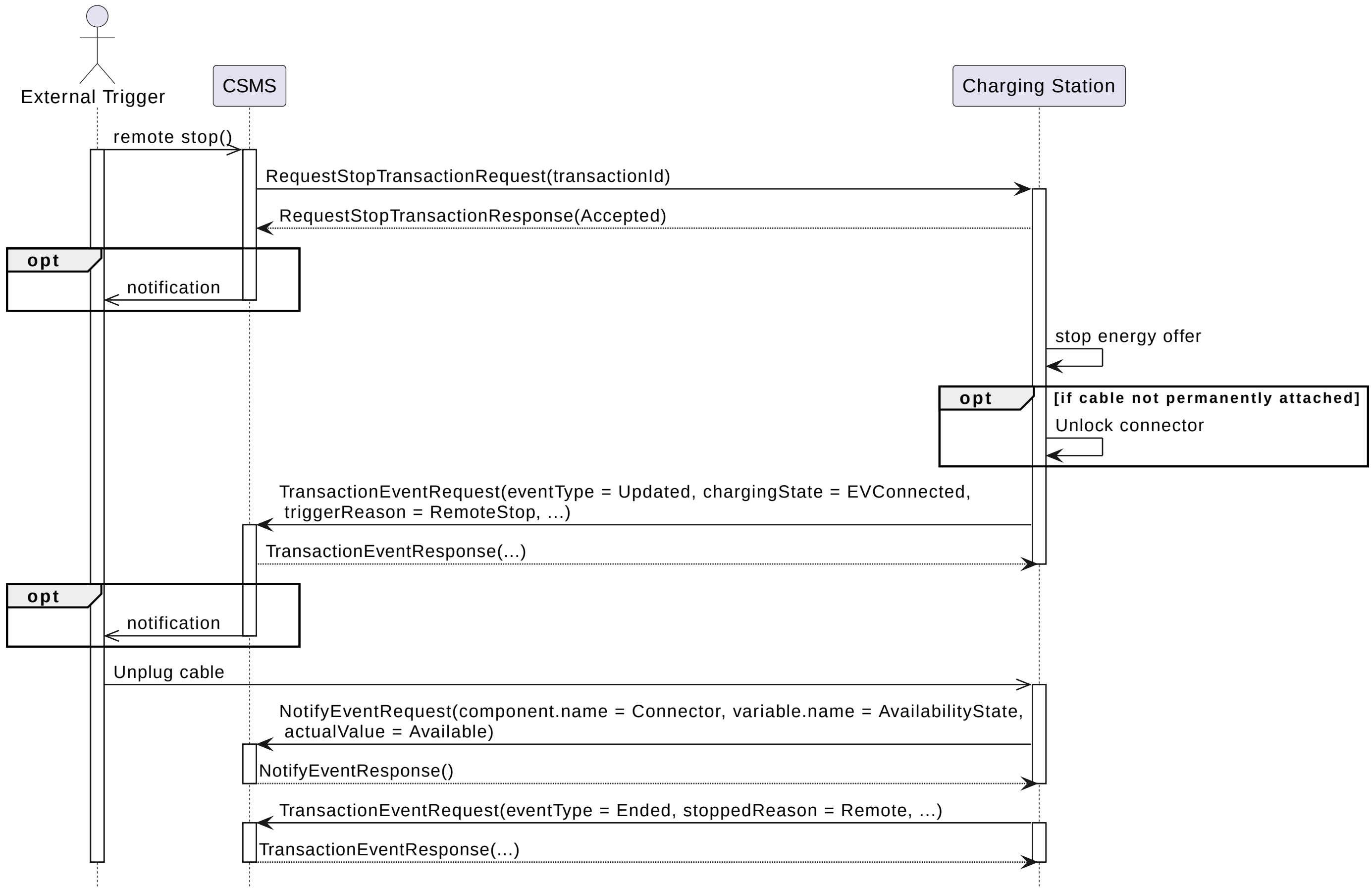

External Trigger
CSMS
Charging Station
remote stop()
RequestStopTransactionRequest(transactionId)
RequestStopTransactionResponse(Accepted)
opt
notification
stop energy offer
opt
[if cable not permanently attached]
Unlock connector
TransactionEventRequest(eventType = Updated, chargingState = EVConnected, triggerReason = RemoteStop, ...)
TransactionEventResponse(...)
opt
notification
Unplug cable
NotifyEventRequest(component.name = Connector, variable.name = AvailabilityState, actualValue = Available)
NotifyEventResponse()
TransactionEventRequest(eventType = Ended, stoppedReason = Remote, ...)
TransactionEventResponse(...)

Figure 81. Sequence Diagram: Remote Stop Transaction

F03 - Remote Stop Transaction - Requirements

Table 79. 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 `Authorized` or `PowerPathClosed`)	The Charging Station SHALL send a TransactionEventRequest (*eventType* = `Ended`, *triggerReason* = `RemoteStop`, *stoppedReason* = `Remote`) to the CSMS.	
F03.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.	Same as E08.FR.09
F03.FR.05	F03.FR.04 AND The Charging Station is running low on memory	The Charging Station MAY drop meter data.	Same as E06.FR.12
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.	Same as E06.FR.13

ID	Precondition	Requirement definition	Note
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 *Accepted*.	
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 *Rejected*.	
F03.FR.09	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.	Same as E06.FR.15

F04 - Remote Stop ISO 15118 Charging from CSMS

No.	Type	Description
1	**Name**	Remote Stop ISO 15118 Charging from CSMS
2	**ID**	F04
	Reference	ISO15118-1 F2 Charging loop with interrupt from the SECC.
3	**Objective(s)**	See ISO15118-1, use case Objective F2, page 38.
4	**Description**	See ISO15118-1, use case Description F2, page 38.
	Actors	EV, CSMS, Charging Station
	Combined scenario description	**OCPP:** **1.** The CSMS sends a RequestStopTransactionRequest to the Charging Station. **2.** The Charging Station responds with a RequestStopTransactionResponse. **ISO 15118-2:** **3.** The EV sends a ChargingStatus (in case of AC charging) or CurrentDemandReq (in case of DC Charging) PDU to the Charging Station. **4.** The Charging Station responds with an EVSENotification = StopCharging. **ISO 15118-20:** **3.** The EV sends a AC_ChargeLoopRes (in case of AC charging) or DC_ChargeLoopRes (in case of DC charging) PDU to the Charging Station. **4.** The Charging Station responds with an EVSENotification = Terminate.
5	**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.
6	**Postcondition(s)**	See ISO15118-1, use case End conditions F2, page 38.
7	**Error handling**	n/a
8	**Remark(s)**	n/a

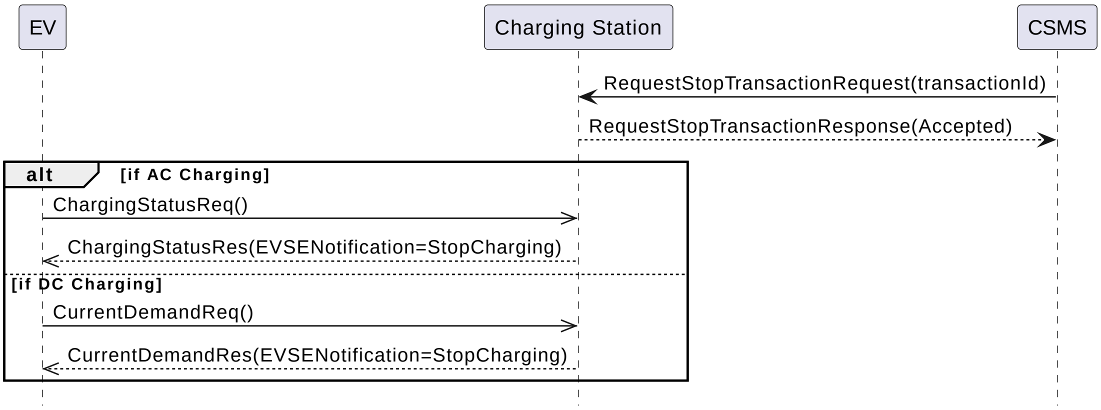

EV
Charging Station
CSMS
RequestStopTransactionRequest(transactionId)
RequestStopTransactionResponse(Accepted)
alt
[if AC Charging]
ChargingStatusReq()
ChargingStatusRes(EVSENotification=StopCharging)
[if DC Charging]
CurrentDemandReq()
CurrentDemandRes(EVSENotification=StopCharging)

Figure 82. Charging loop with interrupt from the Charging Station

F04 - Remote Stop ISO 15118 Charging from CSMS - Requirements

These requirements are normative.

Table 80. 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.	Same as F03.FR.01
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. Same as F03.FR.02
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.	Same as F03.FR.03
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.	Same as E08.FR.09
F04.FR.05	F04.FR.04 AND The Charging Station is running low on memory	The Charging Station MAY drop meter data.	Same as E06.FR.12
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.	Same as E06.FR.13

2.2. Unlock Connector

F05 - Remotely Unlock Connector

No.	Type	Description
1	**Name**	Remotely Unlock Connector
2	**ID**	F05
3	**Objective(s)**	To enable the CSO to help an EV-driver that has problems unplugging his charging cable because the locked 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	**1.** An External Trigger (probably the CSO) request the unlocking of a specific connector of a Charging Station. **2.** The CSMS sends an UnlockConnectorRequest to the Charging Station. **3.** Upon receipt of UnlockConnectorRequest, the Charging Station responds with UnlockConnectorResponse. **4.** 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.
7	**Error handling**	n/a
8	**Remark(s)**	An external trigger, triggering the Unlock command, can be e.g. a Charging Station Operator or an EV Driver app. UnlockConnectorRequest is intended only for unlocking the cable retention lock on the Connector, not for unlocking a Connector access door.

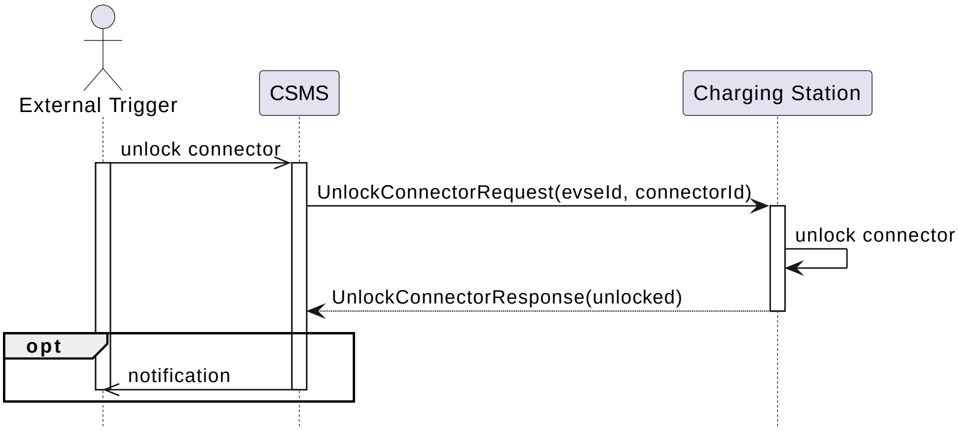


Figure 83. Sequence Diagram: Unlock Connector

F05 - Remotely Unlock Connector - Requirements

Table 81. 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

No.	Type	Description
1	**Name**	Trigger Message
2	**ID**	F06
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	**1.** The CSMS sends a TriggerMessageRequest to the Charging Station. **2.** The Charging Station responds with a TriggerMessageResponse, indicating whether it will send it or not, by returning *Accepted*, *Rejected* or *NotImplemented*. **3.** Message, requested by the CSMS, that the Charging Station marked as *Accepted*, is being sent.
5	**Prerequisite(s)**	The Functional Block *Remote Trigger* is installed.
6	**Postcondition(s)**	**Successful postconditions:** **1.** The CSMS has *Successfully* received a TriggerMessageResponse message. **2.** The CSMS has *Successfully* received a TriggerMessageResponse message with status *Accepted* AND has *Successfully* received the requested message. **Failure postconditions:** **1.** The CSMS has NOT received a TriggerMessageResponse message. **2.** The CSMS has *Successfully* received a TriggerMessageResponse message with status *Accepted* AND has NOT received the requested message.
7	**Error handling**	n/a
8	**Remark(s)**	The TriggerMessage mechanism is not intended to retrieve historic data.

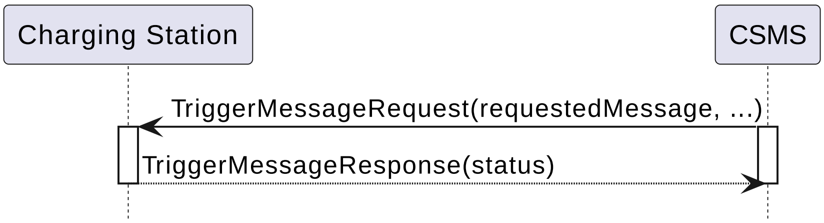

Charging Station
CSMS
TriggerMessageRequest(requestedMessage, ...)
TriggerMessageResponse(status)

Figure 84. Sequence Diagram: Trigger Message

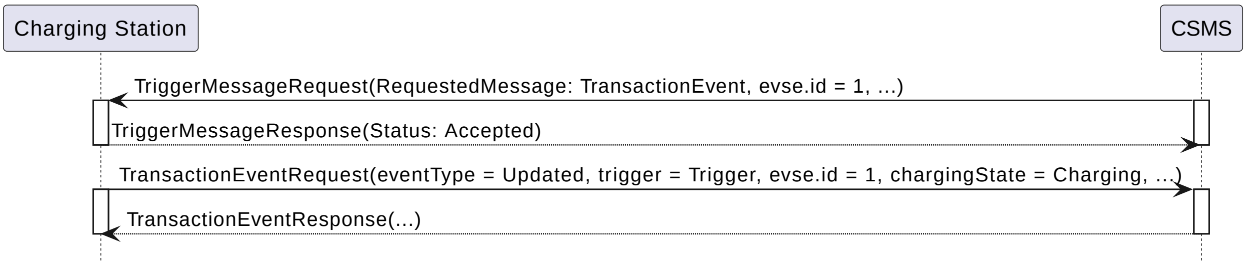

Charging Station
CSMS
TriggerMessageRequest(RequestedMessage: TransactionEvent, evse.id = 1, ...)
TriggerMessageResponse(Status: Accepted)
TransactionEventRequest(eventType = Updated, trigger = Trigger, evse.id = 1, chargingState = Charging, ...)
TransactionEventResponse(...)

Figure 85. Sequence Diagram: Trigger Message Example

F06 - Trigger Message - Requirements

Table 82. 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 evseId 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 TriggerMessage response, 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 accepts 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 accepts 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 MAY respond with a TriggerMessageResponse with status *Rejected*.	StatusNotification messages can only be requested at connector level.

ID	Precondition	Requirement definition	Note
F06.FR.13	When sending a TriggerMessageRequest with *requestedMessage* set to: *StatusNotification*	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 *requestedMessage* set to: *LogStatusNotification* AND The Charging Station is uploading a log file	The Charging Station SHALL send a LogStatusNotificationRequest to the CSMS with status *Uploading*.	
F06.FR.15	If a Charging Station receives a TriggerMessageRequest with *requestedMessage* set to: *LogStatusNotification* AND The Charging Station is NOT uploading a log file	The Charging Station SHALL send a LogStatusNotificationRequest to the CSMS with status *Idle*.	
F06.FR.16	If a Charging Station receives a TriggerMessageRequest with *requestedMessage* set to: *FirmwareStatusNotification* AND The Charging Station is not performing firmware update related tasks.	The Charging Station SHALL send a FirmwareStatusNotificationRequest to the CSMS with status *Idle*.	
F06.FR.17	If Charging Station receives a TriggerMessageRequest with *requestedMessage* set to: *BootNotification* AND the response it received from CSMS to the last BootNotificationRequest was: *Accepted*	Charging Station SHALL respond with a TriggerMessageResponse with status *Rejected*.	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.
F06.FR.18 **(2.1)**	If Charging Station receives a TriggerMessageRequest with *requestedMessage* = `CustomTrigger` AND Charging Station supports the message type in *customTrigger*	Charging Station SHALL reply with *status* = `Accepted` and send the message corresponding to *customTrigger*.	Supported custom triggers are reported in `CustomTriggers`.
F06.FR.19 **(2.1)**	If Charging Station receives a TriggerMessageRequest with *requestedMessage* = `CustomTrigger` AND Charging Station does not support the message type in *customTrigger*	Charging Station SHALL reply with TriggerMessageResponse with *status* = `NotImplemented`.	

F07 - Remote start with fixed cost, energy, SoC or time

New in OCPP 2.1

No.	Type	Description
1	**Name**	Remote start with fixed cost, energy, SoC or time
2	**ID**	F07
3	**Objective(s)**	
4	**Description**	An EV Driver or CSMS specifies a limit in cost, energy, SoC or time. A cost limit is usually related to a prepaid balance or a reservation amount for adhoc payment. An energy or time limit can also be provided.
	Actors	Charging Station, CSMS, EV Driver

No.	Type	Description
	Scenario description	*CSMS specifies a cost limit at remote start* 1. CSMS has a cost limit for the transaction it wants to start on this Charging Station. 2. CSMS sends a RequestStartTransactionRequest to the Charging Station. 3. Charging Station starts a transaction as per use case F01 or F02. 4. In order to communicate the cost limit to the Charging Station, CSMS adds the *transactionLimit.maxCost* element to the first TransactionEventResponse message for this transaction. 5. Charging Station adds *transactionInfo.transactionLimit.maxCost* (once) in the next TransactionEventRequest message. 6. When the cost for energy transfer reaches *transactionInfo.transactionLimit.maxCost*, Charging Station stops the energy transfer and sends a TransactionEventRequest with *chargingState* = SuspendedEVSE.
5	**Prerequisite(s)**	
6	**Postcondition(s)**	Charging Station starts a transaction with given limit(s).
7	**Error handling**	
8	**Remark(s)**	If more than one limit is given, for example both a time and an energy limit, then whichever limit is reached first, determines the end of energy transfer. In order for a Charging Station to stop exactly on time when a cost limit is given, cost will have to be calculated locally at the Charging Station.

NOTE Following diagrams only show parameters related to the use case. More parameters are required in these messages.

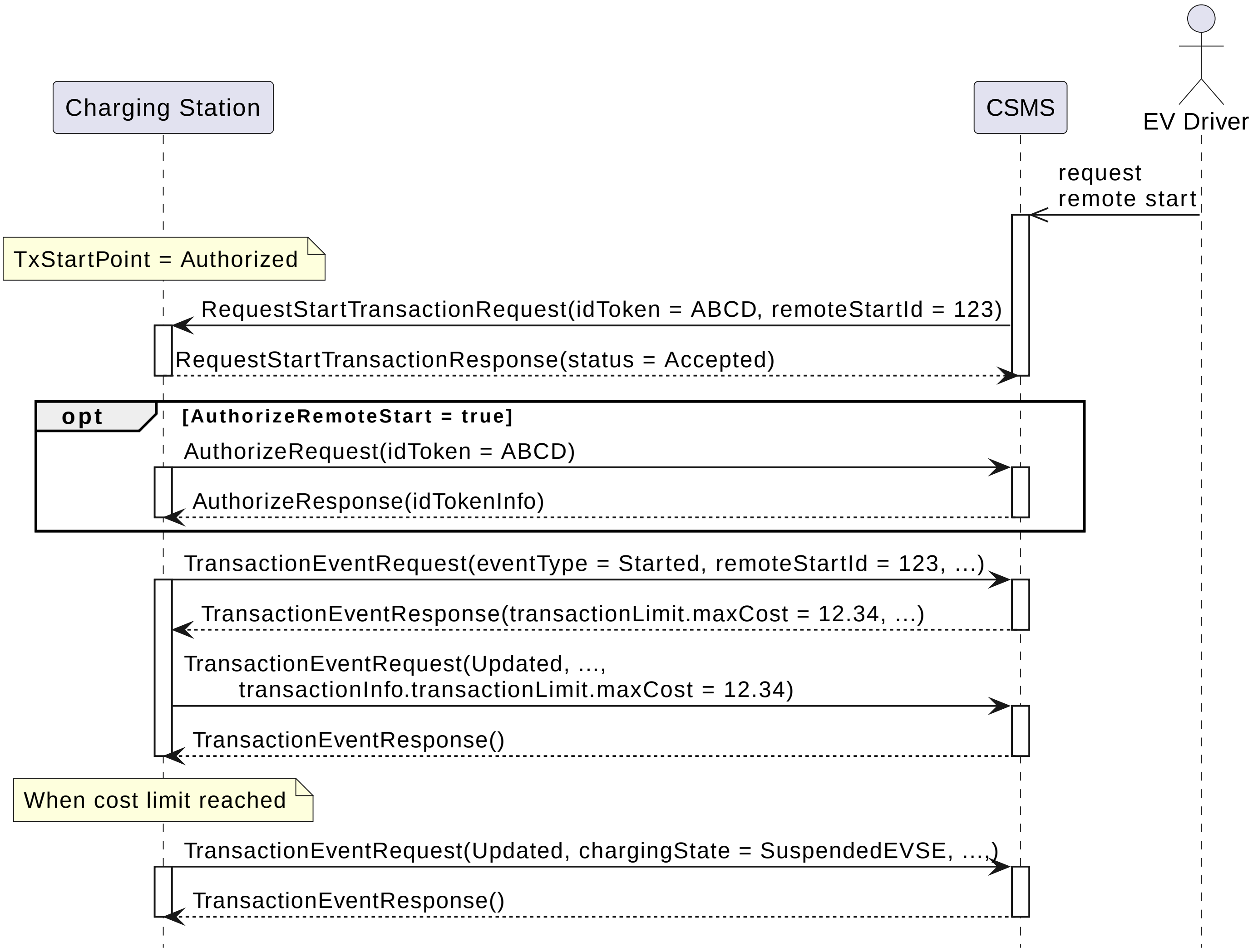

Charging Station
CSMS
EV Driver
request remote start
TxStartPoint = Authorized
RequestStartTransactionRequest(idToken = ABCD, remoteStartId = 123)
RequestStartTransactionResponse(status = Accepted)
opt
[AuthorizeRemoteStart = true]
AuthorizeRequest(idToken = ABCD)
AuthorizeResponse(idTokenInfo)
TransactionEventRequest(eventType = Started, remoteStartId = 123, ...)
TransactionEventResponse(transactionLimit.maxCost = 12.34, ...)
TransactionEventRequest(Updated, ...,
transactionInfo.transactionLimit.maxCost = 12.34)
TransactionEventResponse()
When cost limit reached
TransactionEventRequest(Updated, chargingState = SuspendedEVSE, ...)
TransactionEventResponse()

Figure 86. Sequence diagram for remotely started transaction with limited cost

F07 - Remote start transactions with fixed cost, energy or time - Requirements

ID	Precondition	Requirement definition	Note
F07.FR.01	CSMS wishes to start a transaction with a limit in cost, energy, SoC or time at Charging Station	CSMS SHALL send a RequestStartTransactionRequest message to Charging Station	The limit is sent in the TransactionEvent messages in accordance with E16.
F07.FR.02	F07.FR.01 AND Charging Station sends a TransactionEventRequest with *eventType* = `Started`	CSMS SHALL respond with a TransactionEventResponse with *transactionLimit* set to the required limit.	*transactionLimit* uses *maxCost* for cost limit, *maxEnergy* for energy limit, *maxTime* for time limit or *maxSoC* for SoC limit.
F07.FR.03	F07.FR.02	Charging Station SHALL include this limit in the field *transactionInfo.transactionLimit* (once) in the next TransactionEventRequest	
F07.FR.04		Charging Station SHALL behave as in use case E16 - Transactions with fixed cost, energy, SoC or time.	

NOTE Other requirements from F. Remote Control also apply, notably F01 and F02, as well as E16 - Transactions with fixed cost, energy, SoC or time.

G. Availability

Chapter 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.

Chapter 2. Use cases & Requirements

G01 - Report connector AvailabilityState

No.	Type	Description
1	**Name**	Report connector AvailabilityState
2	**ID**	G01
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	**1.** A Charging Station can send a NotifyEventRequest with *trigger* = `Delta` for *component.name* = "Connector" and the EVSE number in *evse.id* and the connector number in *evse.connectorId*, and *variable* = "AvailabilityState" with the value of the new status to the CSMS. **1a.** Optionally, Charging Station can also include the state of *component* = "ChargingStation" and *component* = "EVSE" in the NotifyEventRequest. **2.** The CSMS responds with NotifyEventResponse to the Charging Station.
	Alternative scenario	**1.** Instead of a NotifyEventRequest a Charging Station can still choose to send the previously used StatusNotificationRequest to the CSMS to inform the CSMS about the new connector status. **2.** The CSMS responds with StatusNotificationResponse to the Charging Station.
5	**Prerequisite(s)**	n/a
6	**Postcondition(s)**	**Successful postconditions:** The CSMS is *Successfully* informed about the status change. **Failure postconditions:** n/a
7	**Error handling**	n/a
8	**Remark(s)**	The Charging Station MAY use the *Unavailable* 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. Notifying a connector status from Charging Station to CSMS via NotifyEventRequest is the preferred method. The old StatusNotificationRequest message is still available for use, but has been deprecated and will be removed in a future release.

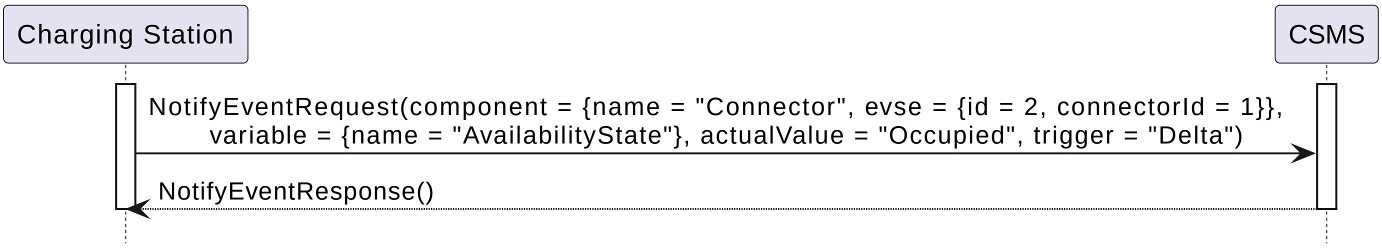

Charging Station
CSMS
NotifyEventRequest(component = {name = "Connector", evse = {id = 2, connectorId = 1}}, variable = {name = "AvailabilityState"}, actualValue = "Occupied", trigger = "Delta")
NotifyEventResponse()

Figure 87. Sequence Diagram: Report connector AvailabilityState

G01 - Report connector AvailabilityState - State transition overview for connecting/disconnecting

Initial	Cable plugin	Cable unplug
Available	→ Occupied	-
Occupied	-	→ Available (→ Unavailable, if scheduled to become Unavailable)
Reserved	-	-

Initial	Cable plugin	Cable unplug
Unavailable	-	-
Faulted	-	-

NOTE In the table below the text: for *component* = "Connector"
should be read as a shorthand notation for:
for *component.name* = "Connector", *component.evse.id* = <evse id>, *component.evse.connectorId* = <connector id>

G01 - Report connector AvailabilityState - Requirements

Table 83. 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 *Unavailable* by a ChangeAvailabilityRequest message.	The EVSE's *Unavailable* status SHALL be persistent across reboots.
G01.FR.03	The connector is `Available` when an EV is connecting	The Charging Station SHALL send a NotifyEventRequest for *component* = "Connector", *variable* = "AvailabilityState", *actualValue* = "Occupied" and *trigger* = "Delta" or a StatusNotificationRequest with *connectorStatus* = `Occupied`.
G01.FR.04	The connector is `Occupied` when an EV is disconnecting AND connector is not scheduled to become `Unavailable` (G03.FR.05)	The Charging Station SHALL send a NotifyEventRequest for *component* = "Connector", *variable* = "AvailabilityState", *actualValue* = "Available" and *trigger* = "Delta" when an EV is disconnected or a StatusNotificationRequest with *connectorStatus* = `Available`.
G01.FR.05	The connector is `Occupied` when an EV is disconnecting AND connector is scheduled to become `Unavailable` (G03.FR.05)	The Charging Station SHALL send a NotifyEventRequest for *component* = "Connector", *variable* = "AvailabilityState", *actualValue* = "Unavailable" and *trigger* = "Delta" when an EV is disconnected or a StatusNotificationRequest with *connectorStatus* = `Unavailable`.
G01.FR.06	The connector is `Reserved` when an EV is connecting AND EV driver presents an IdToken matching the reservation	The Charging Station SHALL send a NotifyEventRequest for *component* = "Connector", *variable* = "AvailabilityState", *actualValue* = "Occupied" and *trigger* = "Delta" or a StatusNotificationRequest with *connectorStatus* = `Occupied`.
G01.FR.07	When a ChangeAvailabilityRequest leads to a connector status change	The Charging Station SHALL send a NotifyEventRequest for *component* = "Connector", *variable* = "AvailabilityState", *trigger* = "Delta" and the corresponding *actualValue* of "AvailabilityState" or a StatusNotificationRequest with the corresponding *connectorStatus*.
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 NotifyEventRequest or StatusNotificationRequest for the other connector(s), even though they are no longer usable.
G01.FR.09	The connector is Reserved when an EV is connecting AND (No IdToken is presented OR EV driver presents an IdToken not matching the reservation)	Connector status SHALL NOT change.

G02 - Heartbeat

No.	Type	Description
1	**Name**	Heartbeat
2	**ID**	G02
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	**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. **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.
5	**Prerequisite(s)**	The heartbeat interval is set.
6	**Postcondition(s)**	**Successful postconditions:**: The CSMS knows the Charging Station is still connected. **Failure postconditions:** The CSMS concludes that the Charging Station is *Offline*.
7	**Error handling**	n/a
8	**Remark(s)**	With JSON over WebSocket, sending heartbeats is *not* 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.

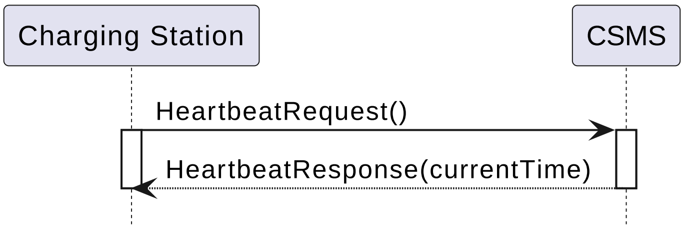


Figure 88. Sequence Diagram: Heartbeat

G02 - Heartbeat - Requirements

Table 84. 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.	
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.	

ID	Precondition	Requirement definition	Note
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

No.	Type	Description
1	**Name**	Change Availability EVSE/Connector
2	**ID**	G03
3	**Objective(s)**	To enable the CSMS to change the availability of an EVSE or Connector to *Operative* or *Inoperative* .
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 *Operative* or *Inoperative*. An EVSE/Connector is considered *Operative* in any status other than *Faulted* and *Unavailable*.
	Actors	Charging Station, CSMS
	Scenario description	**1.** The CSMS sends ChangeAvailabilityRequest requesting a Charging Station to change the availability of an EVSE or Connector. **2.** The Charging Station changes the availability to the EVSE/Connector to the requested operationalStatus from the ChangeAvailabilityRequest. **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. **4**. The Charging Station reports the status of the EVSE/Connector using a NotifyEventRequest.
	Alternative scenario(s)	G04 - Change Availability Charging Station
5	**Prerequisite(s)**	n/a
6	**Postcondition(s)**	**Successful postcondition:** When changing the availability of an EVSE/Connector to *Operative*, the status of the EVSE has changed to *Available*, *Occupied* or *Reserved*. When changing the availability of an EVSE/Connector to *Inoperative* , the status of the EVSE has changed to *Unavailable*. **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.
7	**Error handling**	n/a
8	**Remark(s)**	Persistent states, for example: EVSE set to *Available* SHALL persist a reboot.

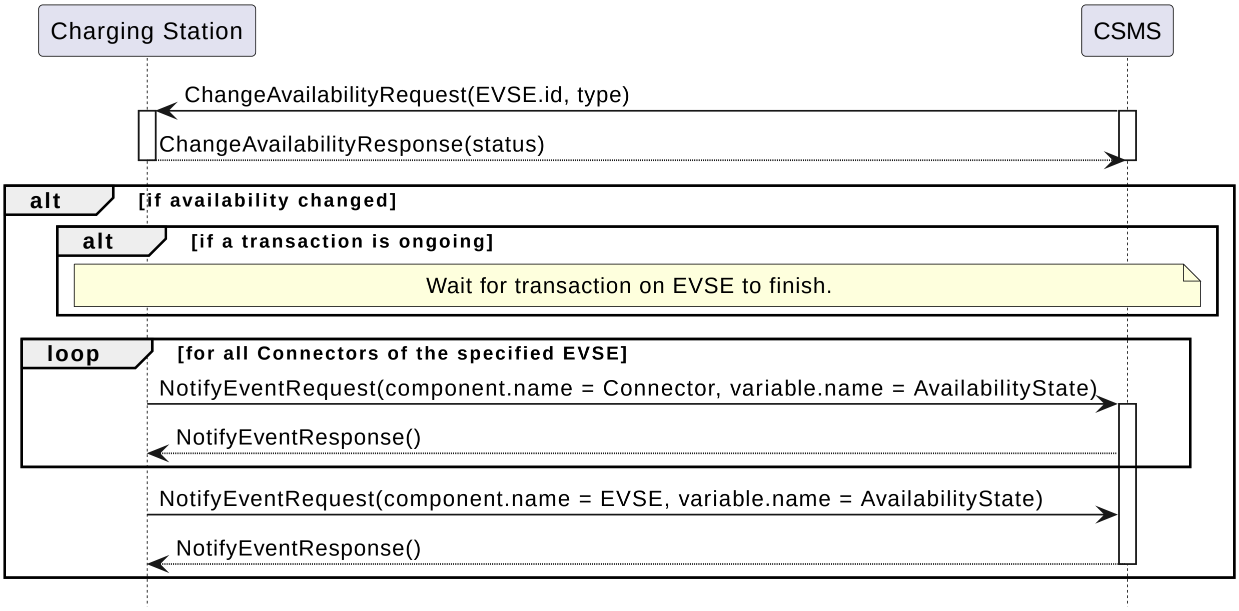

Charging Station
CSMS
ChangeAvailabilityRequest(EVSE.id, type)
ChangeAvailabilityResponse(status)
alt
[if availability changed]
alt
[if a transaction is ongoing]
Wait for transaction on EVSE to finish.
loop
[for all Connectors of the specified EVSE]
NotifyEventRequest(component.name = Connector, variable.name = AvailabilityState)
NotifyEventResponse()
NotifyEventRequest(component.name = EVSE, variable.name = AvailabilityState)
NotifyEventResponse()

Figure 89. Sequence Diagram: Change Availability

G03 - Change Availability EVSE - Requirements

Table 85. 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 or Connector to the state it is already in.	The Charging Station SHALL respond with availability status *Accepted*.	
G03.FR.04	When an availability change request with ChangeAvailabilityRequest has changed the state of a Connector.	The Charging Station SHALL inform the CSMS of its new Connector availability status with StatusNotificationRequest or NotifyEventRequest.	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 *Scheduled* to indicate that it is scheduled to occur after the transaction has finished.	
G03.FR.06	When the availability of an EVSE becomes Inoperative (*Unavailable*, *Faulted*)	All operative connectors (i.e. not *Faulted*) of that EVSE SHALL become *Unavailable*.	
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

No.	Type	Description
1	**Name**	Change Availability Charging Station
2	**ID**	G04
	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**	This use case describes how the CSMS requests the Charging Station to change the availability. A Charging Station is considered *Operative* when it is charging or ready for charging. A Charging Station is considered *Inoperative* when it does *not* allow any charging.
	Actors	Charging Station, CSMS
	Scenario description	**1.** The CSMS sends a ChangeAvailabilityRequest for requesting a Charging Station to change its availability. **2**. Upon receipt of a ChangeAvailabilityRequest, the Charging Station responds with ChangeAvailabilityResponse.
5	**Prerequisite(s)**	n/a
6	**Postcondition(s)**	**Successful postcondition:** The CSMS was able to change the availability of the Charging Station. When changing the availability of a Charging Station to *Operative*, the status of the Charging Station has changed to *Available*. When changing the availability of a Charging Station to *Inoperative*, the status of the Charging Station has changed to *Unavailable*. **Failure postcondition:** The CSMS was *not* able to change the requested Charging Station's availability.
7	**Error handling**	n/a
8	**Remark(s)**	Persistent states: for example, Charging Station set to *Unavailable* SHALL persist a reboot.

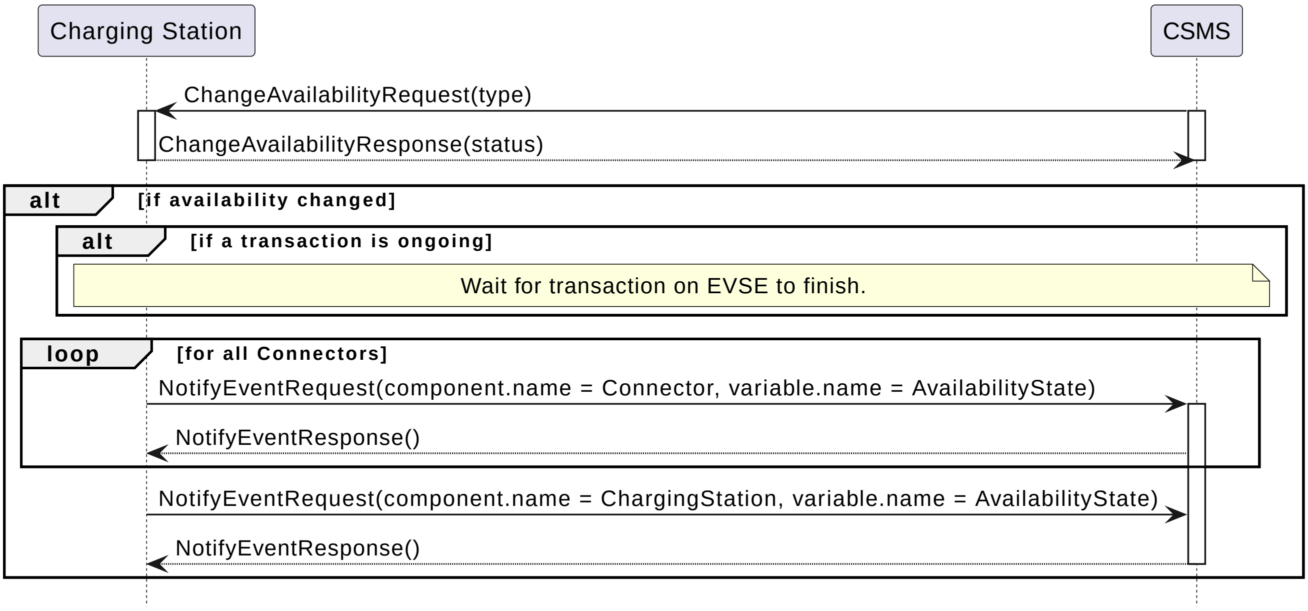

Charging Station
CSMS
ChangeAvailabilityRequest(type)
ChangeAvailabilityResponse(status)
alt
[if availability changed]
alt
[if a transaction is ongoing]
Wait for transaction on EVSE to finish.
loop
[for all Connectors]
NotifyEventRequest(component.name = Connector, variable.name = AvailabilityState)
NotifyEventResponse()
NotifyEventRequest(component.name = ChargingStation, variable.name = AvailabilityState)
NotifyEventResponse()

Figure 90. Sequence Diagram: Change Availability Charging Station

G04 - Change Availability Charging Station - Requirements

Table 86. 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 NotifyEventRequest or 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 (*Unavailable*, *Faulted*)	All operative EVSEs and connectors (i.e. not *Faulted*) SHALL become *Unavailable*.	
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

No.	Type	Description
1	**Name**	Lock Failure
2	**ID**	G05
3	**Objective(s)**	To prevent the EV Driver from charging because 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 because the Connector is not locked properly.
	Actors	Charging Station, CSMS, EV Driver
	Scenario description	**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)**	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.
6	**Postcondition(s)**	Transaction is not started and connector lock event failure is reported.
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"). A lock failure can also be reported when unlocking of the connector fails.

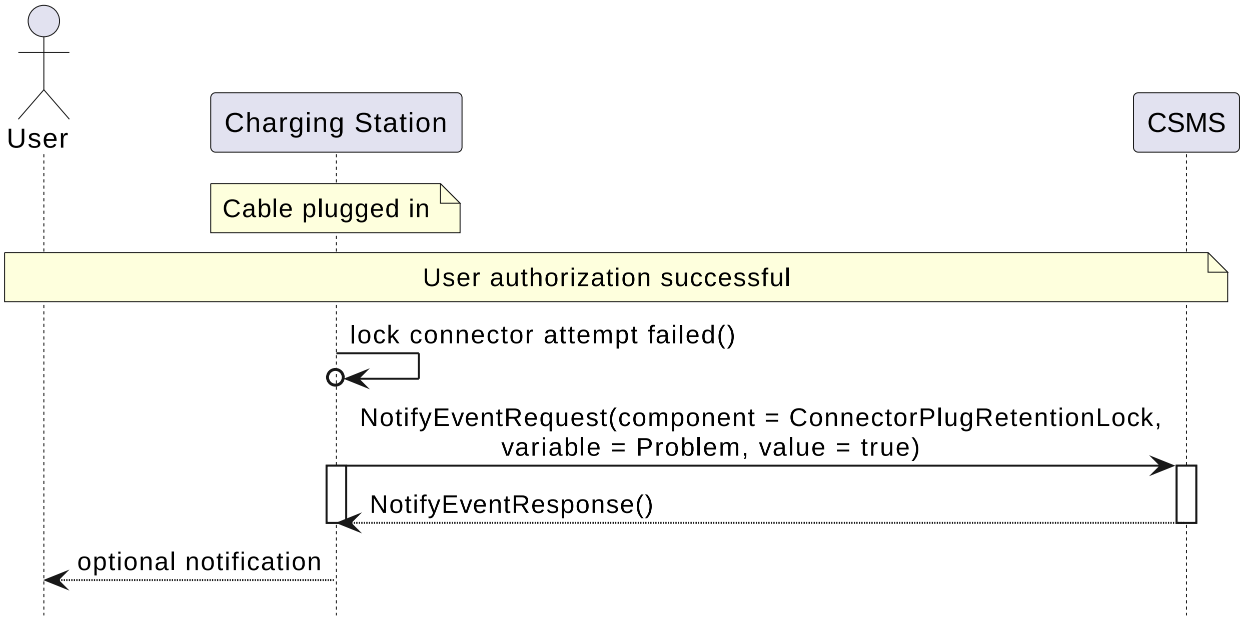

User
Charging Station
CSMS
Cable plugged in
User authorization successful
lock connector attempt failed()
NotifyEventRequest(component = ConnectorPlugRetentionLock, variable = Problem, value = true)
NotifyEventResponse()
optional notification

Figure 91. Sequence Diagram: Lock Failure

G05 - Lock Failure - Requirements

Table 87. 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.	
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

Chapter 1. Introduction

This Functional Block describes the reservation functionality of OCPP. The reservation functionality enables an EV Driver to reserve an EVSE at a Charging Station until a certain time in order to ensure that this EVSE cannot be occupied by another user.

OCPP allows to reserve a specific EVSE at a Charging Station or a specific connector type. The EV Driver can also reserve an unspecified EVSE, in which case the Charging Station will make sure that at least one EVSE remains available for the EV Driver.

Only available EVSEs can be reserved, since a Charging Station cannot know in advance when an occupied EVSE will become available again. This makes it impossible to guarantee a reservation for an EVSE that is currently occupied.

NOTE A CSMS would still be able to support the reservation functionality for occupied EVSEs by delaying the sending of the reservation message to the Charging Station until the EVSE becomes available, but there is no guarantee that it is available in time.

Chapter 2. Use cases & Requirements

H01 - Reservation

No.	Type	Description
1	**Name**	Reservation
2	**ID**	H01
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	**1.** EV Driver asks the CSMS to reserve an unspecified EVSE at the Charging Station. **2.** The CSMS sends ReserveNowRequest without *evseId* 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
	Postcondition(s)	**Successful postcondition:** The Charging Station has accepted the ReserveNowRequest **Failure postcondition:** The Charging Station has rejected the ReserveNowRequest

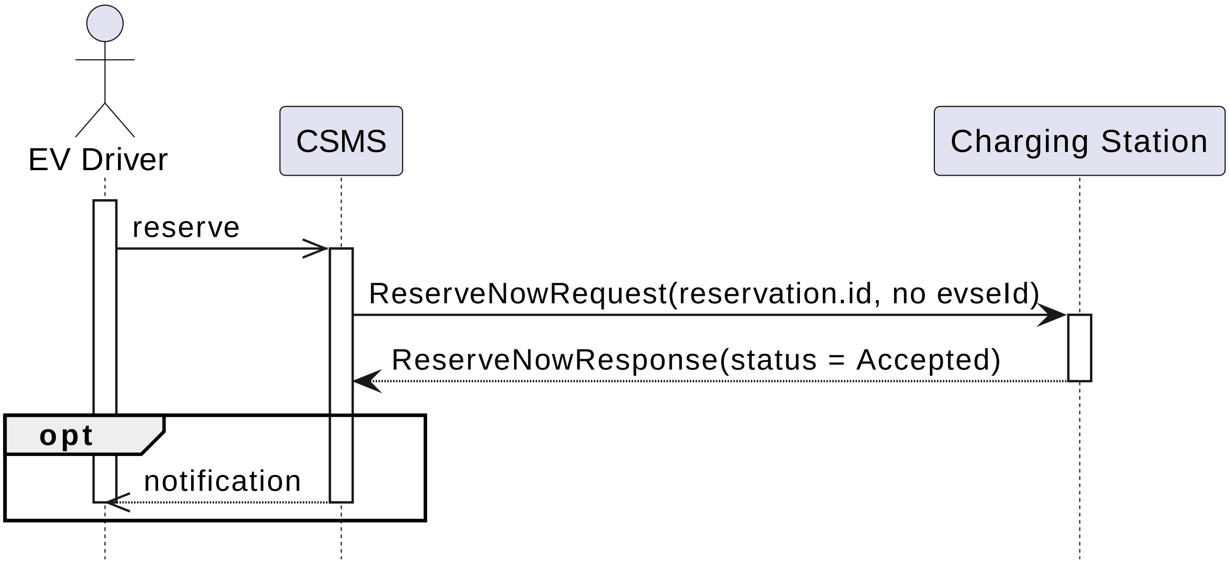

EV Driver
CSMS
Charging Station
reserve
ReserveNowRequest(reservation.id, no evseId)
ReserveNowResponse(status = Accepted)
opt
notification

Figure 92. Sequence Diagram: S1 - Reserve a unspecified EVSE at a Charging Station

S2	*Scenario objective*	Reserve a specific EVSE at a Charging Station
	Scenario description	**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 a NotifyEventRequest with *trigger* = `Delta` for *component.name* = "Connector" and the EVSE number in *evse.id* and the connector number in *evse.connectorId*, *variable* = "AvailabilityState" and *actualValue* = "Reserved".
	Alternative scenario	Steps 1, 2 and 3 as above. **4.** Instead of a NotifyEventRequest a Charging Station can still send the old StatusNotificationRequest with the status *Reserved* for all Connectors of that EVSE.
	Prerequisite(s)	The specified EVSE of the Charging Station has status *Available*

	Postcondition(s)	**Successful postcondition:** The Charging Station has accepted the ReserveNowRequest AND sent NotifyEventRequest with component.name *Connector*, variable.name *AvailabilityState* and actualValue *Reserved* **Failure postcondition:** The Charging Station has rejected the ReserveNowRequest OR The Charging Station has NOT sent NotifyEventRequest with component.name *Connector*, variable.name *AvailabilityState* and actualValue *Reserved*

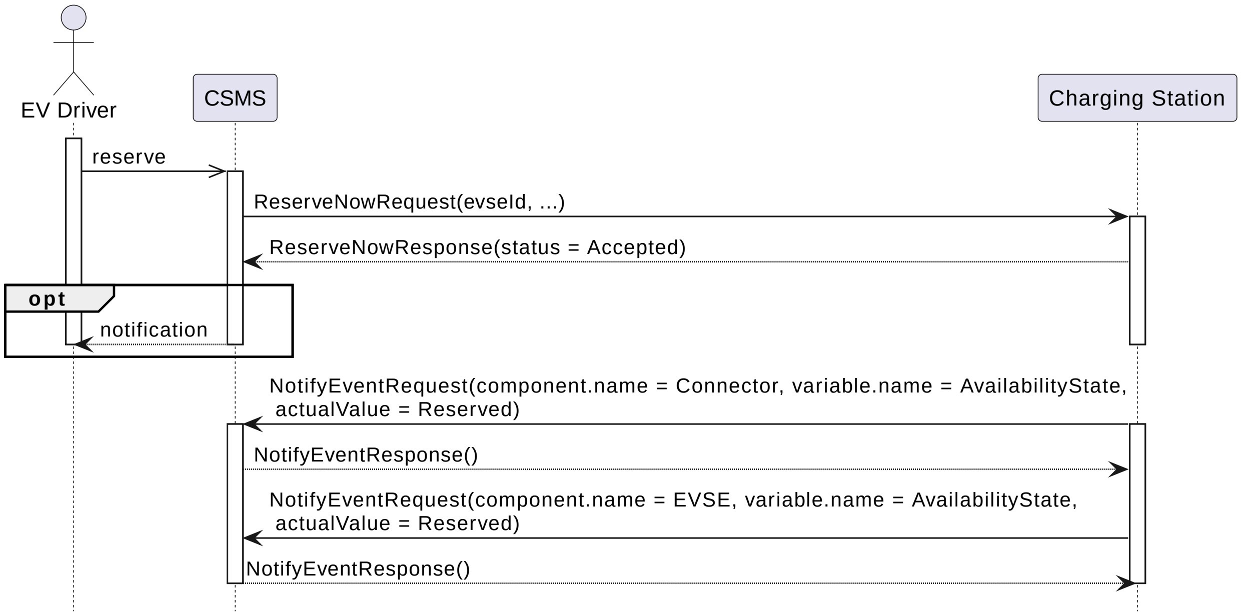

EV Driver
CSMS
Charging Station
reserve
ReserveNowRequest(evseId, ...)
ReserveNowResponse(status = Accepted)
opt
notification
NotifyEventRequest(component.name = Connector, variable.name = AvailabilityState, actualValue = Reserved)
NotifyEventResponse()
NotifyEventRequest(component.name = EVSE, variable.name = AvailabilityState, actualValue = Reserved)
NotifyEventResponse()

Figure 93. 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)	**Successful postcondition:** The Charging Station has accepted the ReserveNowRequest **Failure postcondition:** The Charging Station has rejected the ReserveNowRequest

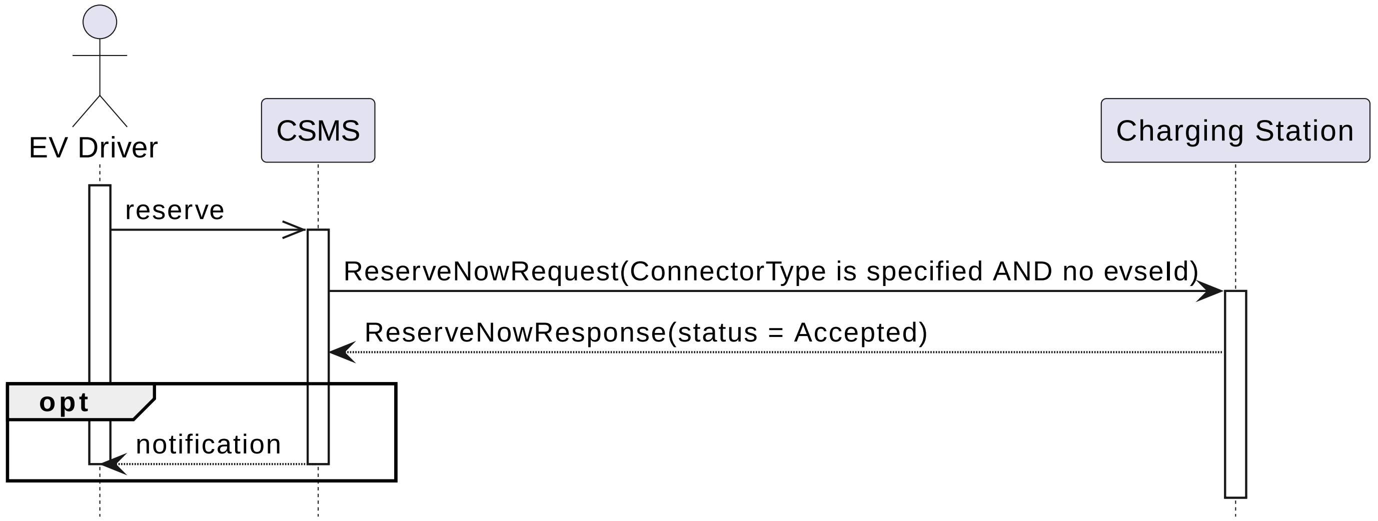

EV Driver
CSMS
Charging Station
reserve
ReserveNowRequest(ConnectorType is specified AND no evseId)
ReserveNowResponse(status = Accepted)
opt
notification

Figure 94. 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 88. 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 `Rejected`.	
H01.FR.02	If the *id* 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 *id* in the ReserveNowRequest does not match any reservation in the Charging Station.	The Charging Station SHALL return the status value `Accepted` if it succeeds in reserving an EVSE.	
H01.FR.04	If the Charging Station receives a ReserveNowRequest without *evseId* AND at least one EVSE is `Available` AND H01.FR.18	The Charging Station SHALL accept the reservation AND respond with a ReserveNowResponse with status `Accepted`.	
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 `Available` AND H01.FR.18	The Charging Station SHALL accept the reservation AND respond with a ReserveNowResponse with status `Accepted`.	
H01.FR.07	When the Charging Station has *Accepted* a ReserveNowRequest without *evseId*	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 `Reserved` or `Occupied`	The Charging Station SHALL return `Occupied`.	
H01.FR.12	When receiving a ReserveNowRequest AND (all) targeted EVSEs have status `Faulted`	The Charging Station SHALL return `Faulted`.	

ID	Precondition	Requirement definition	Note
H01.FR.14	When receiving a ReserveNowRequest AND (all) targeted EVSEs have status `Unavailable`	The Charging Station SHALL return `Unavailable`.	
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 `Faulted`	The Charging Station SHALL cancel the reservation AND send a ReservationStatusUpdate with status `Removed`.	
H01.FR.17	When the status of a targeted EVSE changes to `Unavailable`	The Charging Station SHALL cancel the reservation AND send a ReservationStatusUpdate with status `Removed`.	
H01.FR.18	If the Configuration Variable: `ReservationNonEvseSpecific` is set to *true*.	The Charging Station SHALL accept reservations on an unspecified EVSE.	
H01.FR.19	If the Configuration Variable: `ReservationNonEvseSpecific` is not set or set to *false*.	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 for all connectors of the EVSE: - a NotifyEventRequest with *component* = "Connector", *variable* = "AvailabilityState", *trigger* = "Delta", *actualValue* = "Reserved", OR - a StatusNotificationRequest with *connectorStatus* = `Reserved`.	If an EVSE is reserved, all of its connectors are reported as reserved.
H01.FR.23	If the Charging Station receives a ReserveNowRequest for *evseId* AND this EVSE is `Available`	The Charging Station SHALL respond with a ReserveNowResponse with status `Accepted` AND SHALL send for all connectors of the EVSE: - a NotifyEventRequest with *component* = "Connector", *variable* = "AvailabilityState", *trigger* = "Delta", *actualValue* = "Reserved" OR - a StatusNotificationRequest with *connectorStatus* = `Reserved`.	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 *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 NotifyEventRequest with *component* = "Connector", *variable* = "AvailabilityState", *trigger* = "Delta", *actualValue* = "Reserved" OR - a StatusNotificationRequest with *connectorStatus* = `Reserved`.	If an EVSE is reserved for a specific *connectorType*, all connectors on the EVSE are reported as reserved.

H02 - Cancel Reservation

No.	Type	Description
1	**Name**	Cancel Reservation
2	**ID**	H02
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	**1.** EV Driver asks the CSMS to cancel a reservation. **2.** To cancel a reservation, the CSMS sends CancelReservationRequest to the Charging Station. **3.** If the Charging Station has a reservation matching the reservationId in the request PDU, it returns the status *Accepted*. **4.** If a specific EVSE was reserved for this reservation, the Charging Station sends a NotifyEventRequest with variable "AvailabilityState" set to "Available" for all the Connectors of that EVSE. **5.** The CSMS responds with a NotifyEventResponse to the Charging Station. **6.** The reservation is canceled.
5	**Prerequisite(s)**	- The Functional Block *Reservation* is installed. - EV Driver has a reservation at the Charging Station.
6	**Postcondition(s)**	**Successful postcondition:** The CSMS was able to cancel the EV Driver's reservation at the Charging Stations. **Failure postcondition:** n/a.
7	**Error handling**	n/a
8	**Remark(s)**	The Charging Station does not send a ReservationStatusUpdate, because it was explicitly canceled by CSMS, so it is already aware of the event. Use of StatusNotificationRequest instead of NotifyEventRequest is deprecated, but still allowed.

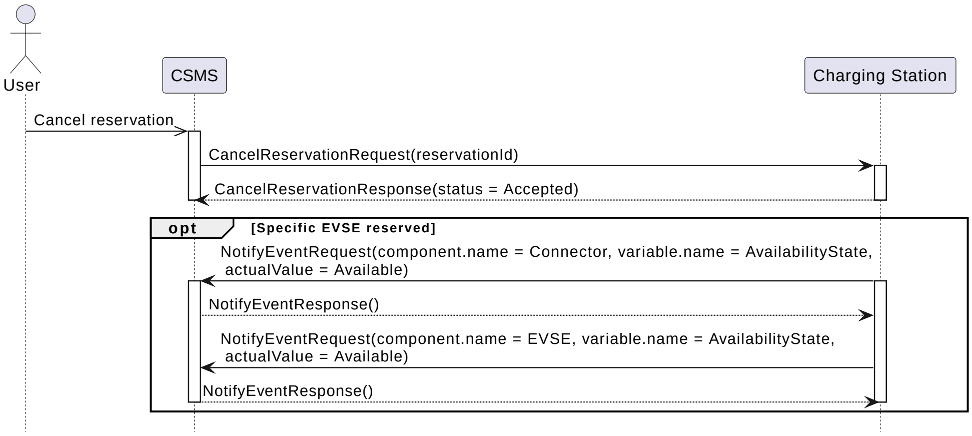

User
CSMS
Charging Station
Cancel reservation
CancelReservationRequest(reservationId)
CancelReservationResponse(status = Accepted)
opt
[Specific EVSE reserved]
NotifyEventRequest(component.name = Connector, variable.name = AvailabilityState, actualValue = Available)
NotifyEventResponse()
NotifyEventRequest(component.name = EVSE, variable.name = AvailabilityState, actualValue = Available)
NotifyEventResponse()

Figure 95. Sequence Diagram: Cancel Reservation

H02 - Cancel Reservation - Requirements

Table 89. 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 *Rejected*.
H02.FR.02	If a Charging Station receives a CancelReservationRequest with a valid, known reservationId.	The reservation SHALL be canceled.

H03 - Use a reserved EVSE

No.	Type	Description
1	**Name**	Use a reserved EVSE
2	**ID**	H03
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 with connector status `Reserved`, that is reserved for this IdToken
	Scenario description	*TxStartPoint = "Authorized"; IdToken presented first* 1. The EV Driver presents an IdTokenType at the Charging Station that is the same as the reservation's IdTokenType. 2. Charging Station matches IdTokenType with the reservation. 3. Connector status becomes `Available`, since reservation has now been consumed. 4. Charging Station optionally authorizes the IdTokenType via an AuthorizeRequest. 5. If authorization accepted, or authorization step was skipped: a. Charging Station starts a transaction as in E03 - Start Transaction - IdToken First. b. Connector status will become `Occupied` when cable is connected.
	Scenario description #2	*TxStartPoint = "EVConnected"; Cable plugged in first* 1. The EV Driver connects the cable. 2. Charging Station starts a transaction, but EVSE connector status remains `Reserved`. 3. The EV Driver presents an IdTokenType at the Charging Station that is the same as the reservation's IdTokenType 4. Charging Station matches IdTokenType with the reservation 5. Connector status becomes `Occupied`, since reservation has now been consumed 6. Charging Station optionally authorizes the IdTokenType via an AuthorizeRequest 7. If authorization accepted, or authorization step was skipped: a. Charging Station starts a transaction as in E02 - Start Transaction - Cable Plugin First
5	**Prerequisite(s)**	EVSE has been reserved for IdToken and connector status is `Reserved`.
6	**Postcondition(s)**	n/a

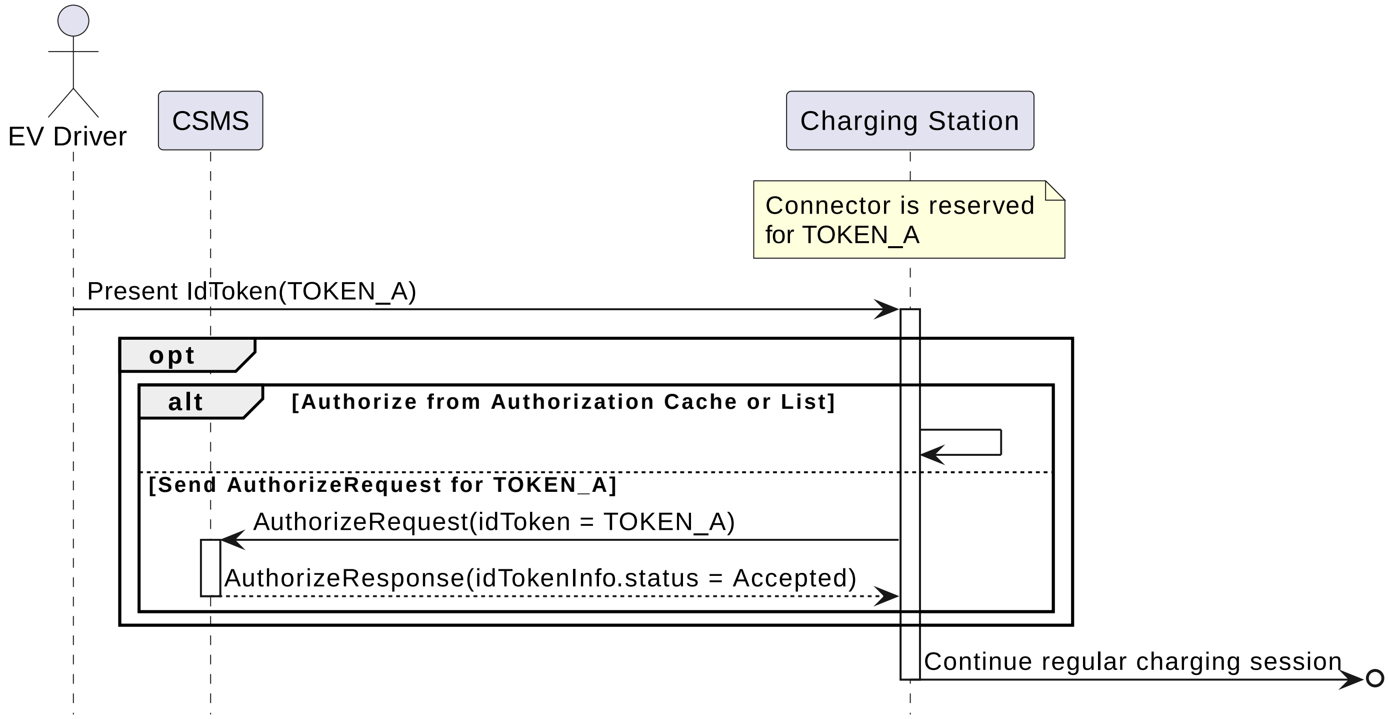

EV Driver
CSMS
Charging Station
Connector is reserved for TOKEN_A
Present IdToken(TOKEN_A)
opt
alt
[Authorize from Authorization Cache or List]
[Send AuthorizeRequest for TOKEN_A]
AuthorizeRequest(idToken = TOKEN_A)
AuthorizeResponse(idTokenInfo.status = Accepted)
Continue regular charging session

Figure 96. Sequence Diagram: Use a reserved EVSE with IdToken

S2	*Scenario objective*	Use an EVSE with connector status `Reserved`, that is reserved for this GroupIdToken
	Scenario description	*TxStartPoint = "Authorized"; IdToken presented first* 1. The EV Driver presents an IdTokenType at the Charging Station that is **not** the same as the reservation's IdTokenType, but the reservation contains a *groupIdToken*. 2. Charging Station authorizes the IdTokenType via an AuthorizeRequest, Local Authorization List or Authorization Cache, and checks if the *groupIdToken* of the IdTokenType matches with the reservation. 3. If *groupIdTokens* match: a. Connector status becomes `Available`, since reservation has now been consumed. b. Charging Station starts a transaction as in E03 - Start Transaction - IdToken First c. Connector status will become `Occupied` when cable is connected
	Scenario description #2	*TxStartPoint = "EVConnected"; Cable plugged in first* 1. The EV Driver connects the cable. 2. Charging Station starts a transaction, but connector status remains `Reserved`. 3. The EV Driver presents an IdTokenType at the Charging Station that is **not** the same as the reservation's IdTokenType, but the reservation contains a *groupIdToken*. 4. Charging Station authorizes the IdTokenType via an AuthorizeRequest, Local Authorization List or Authorization Cache, and checks if the *groupIdToken* of the IdTokenType matches with the reservation. 5. If *groupIdTokens* match: a. Connector status becomes `Occupied`, since reservation has now been consumed. b. Charging Station starts a transaction as in E02 - Start Transaction - Cable Plugin First
5	**Prerequisite(s)**	EVSE has been reserved for GroupIdToken. EVSE *connectorStatus* = `Reserved`.
6	**Postcondition(s)**	n/a

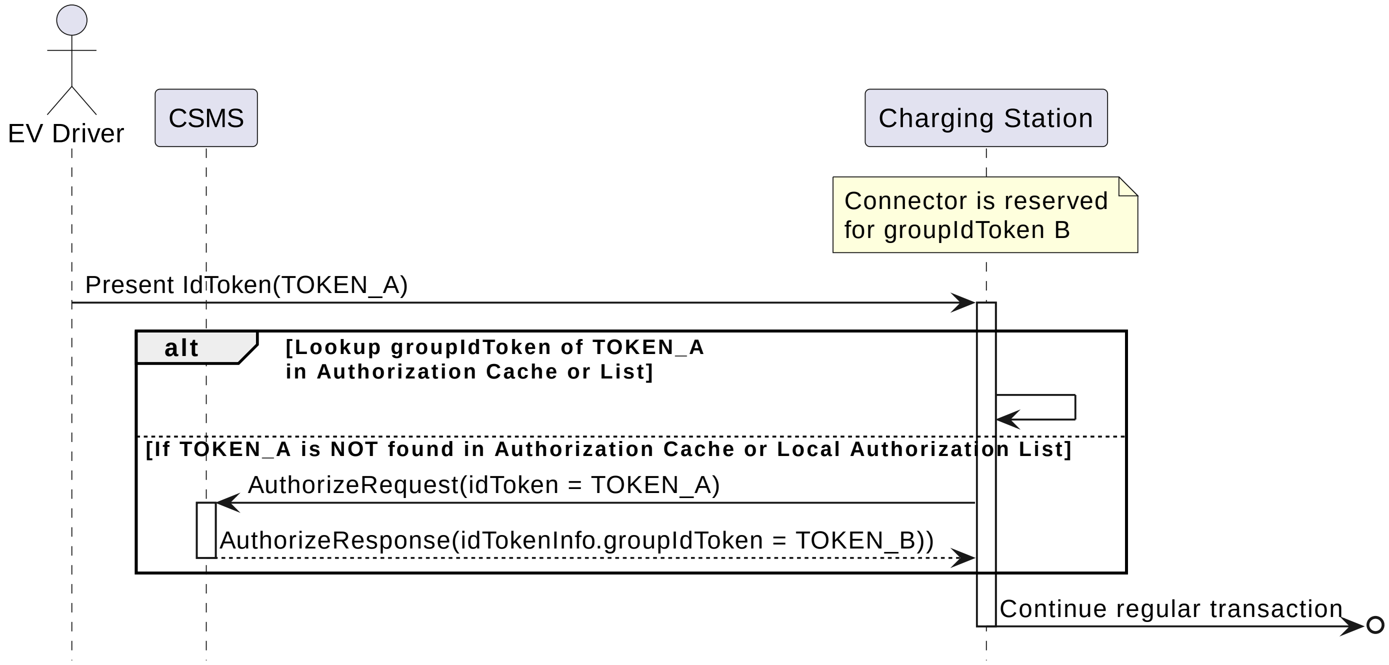

EV Driver
CSMS
Charging Station
Connector is reserved for groupIdToken B
Present IdToken(TOKEN_A)
alt
[Lookup groupIdToken of TOKEN_A in Authorization Cache or List]
[If TOKEN_A is NOT found in Authorization Cache or Local Authorization List]
AuthorizeRequest(idToken = TOKEN_A)
AuthorizeResponse(idTokenInfo.groupIdToken = TOKEN_B))
Continue regular transaction

Figure 97. Sequence Diagram: Use a reserved EVSE with GroupId

S3	*Scenario objective*	Use an EVSE when Charging Station has a reservation for idToken, but connector status is `Available`. This happens when reservation is for an unspecified EVSE and multiple EVSEs are available.
	Scenario description	*TxStartPoint = "Authorized"; IdToken presented first* Identical to scenario S1 above.
	Scenario description #2	*TxStartPoint = "EVConnected"; Cable plugged in first* 1. The EV Driver connects the cable 2. Charging Station reports connector status as `Occupied` 3. Charging Station starts a transaction 4. The EV Driver presents an IdTokenType at the Charging Station that is the same as the reservation's IdTokenType 5. Charging Station matches IdTokenType with the reservation 6. Charging Station optionally authorizes the IdTokenType via an AuthorizeRequest 7. If authorization accepted, or authorization step was skipped: a. Charging Station starts a transaction as in E02 - Start Transaction - Cable Plugin First
5	**Prerequisite(s)**	Unspecified EVSE has been reserved for idToken. EVSE connector status is `Available`.
6	**Postcondition(s)**	n/a

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. If an *idToken* is presented that does not match the reservation (and *groupIdTokens* do not match either), then this *idToken* is not authorized to charge. If TxStartPoint = `Authorized` or `PowerPathClosed` then a transaction would not be started in this case. If TxStartPoint = `EVConnected` or `ParkingBayOccupancy` then a transaction would be started by cable plug-in or occupancy of parking bay, but charging would not start. Assuming a TxStopPoint of `EVConnected` the transaction would be ended at cable plug-out.

H03 - Use a reserved EVSE - Requirements

Table 90. H03 - Requirements

ID	Precondition	Requirement definition	Note
H03.FR.01	Reservation is pending for a specific *idToken* for a specific *evseId*	The Charging Station SHALL allow charging on that EVSE when IdToken presented for authorization matches the specific *idToken* from the reservation.	
H03.FR.02	Reservation is pending for a specific *idToken* for a specific *connectorType*	The Charging Station SHALL allow charging on an EVSE with a connector of type *connectorType* when IdToken presented for authorization matches the specific *idToken* from the reservation.	
H03.FR.03	Reservation is pending for a specific *idToken* without a specific *evseId* or *connectorType*	The Charging Station SHALL allow charging on an EVSE when IdToken presented for authorization matches the specific *idToken* from the reservation.	
H03.FR.04	H03.FR.01 AND attribute *groupIdToken* in reservation has a value	The Charging Station SHALL allow charging on that EVSE when IdToken presented for authorization matches the specific *idToken* from the reservation or when the associated *groupIdToken* matches.	
H03.FR.05	H03.FR.02 AND attribute *groupIdToken* in reservation has a value	The Charging Station SHALL allow charging on an EVSE with a connector of type *connectorType* when IdToken presented for authorization matches the specific *idToken* from the reservation or when the associated *groupIdToken* matches.	
H03.FR.06	H03.FR.03 AND attribute *groupIdToken* in reservation has a value	The Charging Station SHALL allow charging on any EVSE when IdToken presented for authorization matches the specific *idToken* from the reservation or when the associated *groupIdToken* matches.	
H03.FR.07	If attribute *groupIdToken* in the reservation has a value (it is optional).	In order to determine the *groupIdToken* that is associated with an incoming IdToken, the Charging Station MAY look it up in its Local Authorization List or Authorization Cache.	
H03.FR.08	H03.FR.07 AND If the incoming IdToken is not found in the Local Authorization List or Authorization Cache.	The Charging Station SHALL send an AuthorizeRequest for the incoming IdToken to the CSMS in order to get its associated *groupIdToken*. (Note: This AuthorizeRequest may already have been performed when the idToken was presented for authorization.)	
H03.FR.09	When an *idToken* or *groupIdToken* is presented that matches a reservation	Charging Station SHALL consider the reservation to be used (consumed)	The *(group)IdToken* can be presented locally at a card reader, but can also be part of a RequestStartTransaction .
H03.FR.10	H03.FR.09 AND Connector associated with reservation has status `Reserved`	Charging Station SHALL set connector status to `Available` if no cable has been plugged-in, or `Occupied` if a cable has already been plugged-in.	
H03.FR.11 **(2.1)**	When an *idToken* or *groupIdToken* is presented that matches a reservation AND No transaction is started within EVConnectionTimeout	The Charging Station IS RECOMMENDED to send a ReservationStatusUpdateRequest with *status* = `NoTransaction`.	This can happen when cable is never plugged in and TxStartPoint is EVConnected or PowerPathClosed.

H04 - Reservation Ended, not used

No.	Type	Description
1	**Name**	Reservation Ended, not used
2	**ID**	H04
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	**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 NotifyEventRequest with variable "AvailabilityState" set to "Available" for all the Connectors of that EVSE. **5.** The CSMS responds with a NotifyEventResponse. **6.** The Charging Station sends a ReservationStatusUpdateRequest with status *Expired* to the CSMS. **7.** The CSMS responds with a ReservationStatusUpdateResponse.
5	**Prerequisite(s)**	n/a
6	**Postcondition(s)**	n/a
7	**Error handling**	n/a
8	**Remark(s)**	Use of StatusNotificationRequest instead of NotifyEventRequest is deprecated, but still allowed.

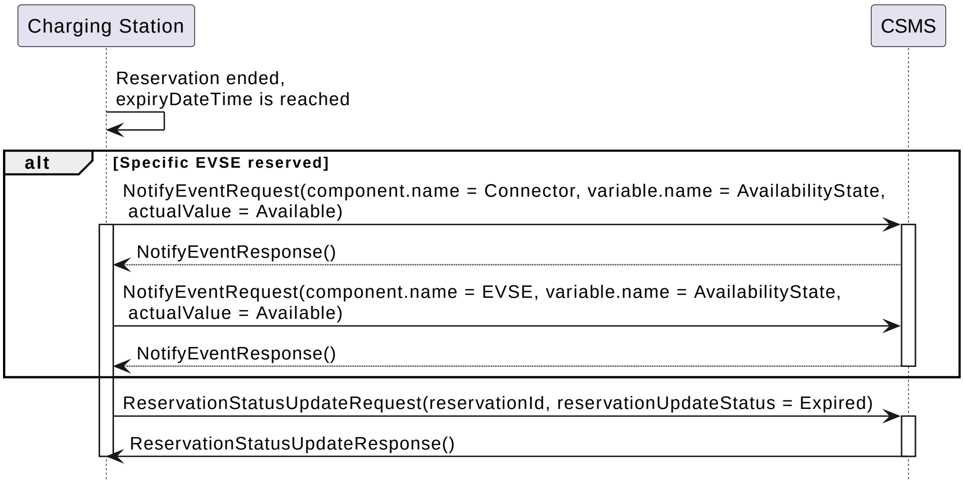

Charging Station
CSMS
Reservation ended,
expiryDateTime is reached
alt
[Specific EVSE reserved]
NotifyEventRequest(component.name = Connector, variable.name = AvailabilityState,
actualValue = Available)
NotifyEventResponse()
NotifyEventRequest(component.name = EVSE, variable.name = AvailabilityState,
actualValue = Available)
NotifyEventResponse()
ReservationStatusUpdateRequest(reservationId, reservationUpdateStatus = Expired)
ReservationStatusUpdateResponse()

Figure 98. Sequence Diagram: Reservation Ended, not used

H04 - Reservation Ended, not used - Requirements

Table 91. H04 - Requirements

ID	Precondition	Requirement definition
H04.FR.01	The reservation ends (*expiryDateTime* reached)	The Charging Station SHALL send a ReservationStatusUpdateRequest with status *Expired*.
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 *Available* or a NotifyEventRequest with *AvailabilityState* set to *Available* to the CSMS for each connector, notifying the CSMS that all the connectors of this EVSE are available again for any EV Driver.

I. Tariff And Cost

Chapter 1. Introduction

Updated in OCPP 2.1

This Functional Block specifies how to provide tariff and cost information to an EV Driver, when a Charging Station is capable of showing this on a display.

Earlier versions of OCPP only allowed for central cost calculation to be performed by CSMS. OCPP 2.1 adds the possibility to provide a tariff plan to the Charging Station to enable it do local cost calculation. Local cost calculation adds complexity to the Charging Station, but it offers the advantage that it is easy to provide a real-time running cost display, and it lets you stop the charge at a preset amount (e.g. for prepaid charging).

The OCPP tariff structure that describes the price structure to use for local cost calculation, is a list of components, like energy and time, with restrictions describing when they apply, like time of day, power, etc. A similar format is used in the common roaming protocols, OCPI (peer-to-peer, Gireve), OCHP (e-clearing.net) and OICP (Hubject). OCPI and OCHP both use tariff elements with components and restrictions. OICP uses pricing products with product availability times and a maximum charging power. All three formats can be mapped straightforwardly onto the OCPP tariff format.

1.1. Configuration Settings

Support for local cost calculation is controlled via the TariffCostCtrlr component. Local cost calculation is enabled by setting TariffCostCtrlr.Enabled[Tariff] to true (see `TariffEnabled`). The frequency by which variations in current, power and energy need to be monitored for tariff calculation is determined by TariffCostCtrlr.Interval[Tariff] (see `TariffInterval`). The maximum supported complexity of a tariff can be limited by TariffCostCtrlr.MaxElements[Tariff] (see `TariffMaxElements`).

The sending of intermediate running cost updates is enabled by setting TariffCostCtrlr.Enabled[Cost] to true (see `CostEnabled`). How frequent running cost updates must be sent, is determined by TariffCostCtrlr.Interval[Cost] (see `CostInterval`).

NOTE Charging Station stores a driver-specific tariff, that is received in an AuthorizeResponse, in the Authorization Cache. A CSO is recommended to either disable the use of an Authorization Cache or clear the authorization cache whenever new tariffs become available.

WARNING A Local Authorization List cannot be used in conjunction with local cost calculation, because an AuthorizeRequest is needed to get the driver-specific tariff. Only when the default tariff applies to all drivers can a Local Authorization List be used.

1.2. Tariff Structures

A tariff is described by the TariffType, which consists of fields with prices for:

- energy,
- charging time,
- idle time,
- fixed fee.

Each of these fields may have (optional) conditions that specify when a price is applicable. A reservation is billed as a fixed fee or based on time.

The energy field is of type TariffEnergyType which has a TariffEnergyPriceType with a price per kWh for the energy and an optional TariffConditionsType (date, time of day, power, current, energy) that describes under which conditions this TariffEnergyPriceType is to be applied.
The charging time and idle time fields are of type TariffTimeType which has a TariffTimePriceType with a price per minute for the time and an optional TariffConditionsType.
Similarly, a fixed fee is described by a TariffFixedType with an optional TariffConditionsFixedType, which does not contain conditions for energy, power, current or time, since it is applied at the start of a transaction.

A Tariff{Energy/Time/Fixed}PriceType without a TariffConditions(Fixed)Type is always applicable. In case multiple prices are applicable, then only the first applicable element in the list is applied.

A Charging Station implementation can opt to not support the TariffConditions(Fixed)Type element. If this is the case, then it must report TariffCostCtrlr.ConditionsSupported[Tariff] as false, so that CSMS is aware that it cannot use conditions in tariff plans.

1.2.1. Simple kWh tariff

The following is a basic example of a JSON tariff structure that describes a kWh-based price for a transaction:

Simple kWh tariff of $0.25 per kWh

```
{
  "id": "10",
  "currency": "USD",
  "energy": {
      "taxRates": [
        { "type": "federal", "tax": 6.0 },
        { "type": "state", "tax": 4.0 }
      ],
      "prices": [{
        "priceKwh": 0.25
    }]
  }
}
```

1.2.2. Time-of-day kWh tariff with idle fee

The following example describes a tariff with a higher price during office hours (8-18h) and an idle fee for staying connected without charging, and a lower price without idle fee during evening and night.

Time-of-day kWh tariff with idle fee

```
{
    "id": "11",
    "currency": "EUR",
    "energy": {
        "taxRates": [
            {
                "type": "vat",
                "tax": 4
            }
        ],
        "prices": [
            {
                "priceKwh": 0.4,
                "conditions": {
                    "startTimeOfDay": "08:00",
                    "endTimeOfDay": "18:00"
                }
            },
            {
                "priceKwh": 0.25
            }
        ]
    },
    "idleTime": {
        "taxRates": [
            {
                "type": "vat",
                "tax": 4
            }
        ],
        "prices": [
            {
                "priceMinute": 1,
                "conditions": {
                    "startTimeOfDay": "08:00",
                    "endTimeOfDay": "18:00"
                }
            }
        ]
    }
}
```

NOTE

startTimeOfDay and *endTimeOfDay* in TariffConditions(Fixed)Type are in local time (without time zone). Tariff shows times as presented to driver. This is always local time. This also avoids the complexities of separate tariffs for summer and winter time and how to deal with switching between them. A Charging Station can easily convert times to the internally used time zone. This is not required to be UTC, as described in Section 3.1: "It is strongly RECOMMENDED to exchange all time values between CSMS and Charging Station as UTC"

1.2.3. Complex hourly tariff

The following describes a time-based tariff, that includes a higher start fee for credit cards and an hourly fee that depends on time of the day and day of the week.

Complex hourly tariff with conditions based on time and day of week and an idle fee

```
{
  "id": "12",
  "currency": "EUR",
  "fixedFee": {
    "taxRates": [ { "type": "vat", "tax": 10.0 } ],
    "prices": [{
        "priceFixed": 3.00,     -- start fee for credit card
        "conditions": {
            "paymentRecognition": "CC"
        }
    },
    {
        "priceFixed": 2.50     -- start fee other payment methods
    }]
  },
  "chargingTime": {
      "taxRates": [ { "type": "vat", "tax": 15.0 } ],
      "prices": [{
        "priceMinute": 1.00,    -- price per minute
        "conditions": {
            "maxPower": 11000      -- when charging speed < 11.0 kW.
        }
      },
      {
        "priceMinute": 2.00,
        "conditions": {
            "minPower": 11000       -- when charging speed >= 11.0 kW
        }
      }]
  },
  "idleTime": {
      "taxRates": [ { "type": "vat", "tax": 15.0 } ],
      "prices": [{
        "priceMinute": 1.00,    -- price per minute
        "conditions": {
          "startTimeOfDay": "09:00",
          "endTimeOfDay": "18:00",
          "minIdleTime": 300,   -- after 5 mins of idle time
          "dayOfWeek": ["Monday", "Tuesday", "Wednesday", "Thursday",
"Friday"]
        }
      },
      {
        "priceMinute": 0.60,    -- price per minute
        "conditions": {
          "startTimeOfDay": "10:00",
          "endTimeOfDay": "17:00",
          "dayOfWeek": ["Saturday"]
        } -- No idle fee on other days (Sunday) or hours
      }]
  }
}
```

NOTE When tariff conditions are used, it is recommended to add a field without any condition at the bottom of the list as a fallback to use when none of the conditions are met.

1.2.4. Idle fee based on charging location occupancy

When an operator only wants to charge an idle fee when the charging location is occupied for more than a certain percentage, then this is something that cannot be captured in a static tariff, since one charging station is not aware of activity at other charging stations. Therefore, CSMS will update the tariff to use, depending on the occupancy of the charging location.

When the charging location is quiet, for example less than half of the station is in use, CSMS uses the following tariff:

kWh tariff for low occupancy (no idle fee)

```
{
  "id": "21",
  "description": [{
    "format": "ASCII", "language": "en",
    "content": "0.44 ct/kWh (currently no idle fees, but will apply at busy
times, 5 minutes after end of charging). Price incl. VAT"
  }],
  "currency": "EUR",
  "energy": [{
    "prices": [{
      "pricekWh": 0.40,       -- per kWh
      "taxRates": [ { "type": "vat", "tax": 10.0 } ]
    }]
  }]
}
```

At the moment that more than half of the stations at this location are in use, CSMS will use below tariff for new charging sessions, but also update the tariff for existing charging sessions with this tariff. The kWh price remains the same, but idle time will now be charged after 5 minutes.

kWh tariff for high occupancy (with idle fee)

```
{
  "id": "22",
  "description": [{
    "format": "ASCII", "language": "en",
    "content": "0.44 ct/kWh (idle fees of 1 euro per minute apply 5 minutes
after end of charging, until location becomes less busy). Price incl. VAT"
  }],
  "currency": "EUR",
  "energy": [{
    "prices": [{
      "priceKwh": 0.40,    -- per kWh
      "taxRates": [ { "type": "vat", "tax": 10.0 } ]
    }]
  }, {
    "idleTime": [{
      "priceMinute": 0.909,
      "taxRates": [ { "type": "vat", "tax": 10.0 } ]
    }],
    "conditions": {
      "minIdleTime": 300    -- 5 minutes
    }
  }]
}
```

The change of tariff will be reflected in the CostDetailsType as a new ChargingPeriodType, and it will be reported in the transaction events as a *triggerReason* = `TariffChanged`.

1.3. CostDetails

At the end of a transaction the Charging Station includes a break-down of the calculated cost in the last TransactionEventRequest in a CostDetailsType.

The CostDetailsType contains a list of ChargingPeriodType that each represent a period during the transaction to which the same TariffEnergyPriceType and [tarifftimepricetype] was applied. Whenever a different TariffEnergyPriceType or [tarifftimepricetype] becomes valid, because conditions change and a different TariffConditionsType applies, a new ChargingPeriodType is created. Note, that a TariffFixedPriceType is only evaluated at start of the transaction.

A simple energy or time-based tariff without conditions will result in a CostDetailsType with only one ChargingPeriodType.

CostDetails example for tariff 10 (see above) when charging 10 kWh

```
{
  "chargingPeriods": [{
    "tariffId": "10",
    "startPeriod": "2023-04-05T14:01:02Z",
    "dimensions": [{
      "type": "energy", -- energy is relevant dimension for tariff
      "volume": 10000
    }]
  }],
  "totalCost": {
    "currency": "EUR",
    "energy": {
      "exclTax": 2.50,
      "inclTax": 2.75,
      "taxRates": [
        { "type": "federal", "tax": 6.0 },
        { "type": "state", "tax": 4.0 }
      ]
    },
    "total": {
      "exclTax": 2.50,
      "inclTax": 2.75
    }
  }
  "totalUsage": {
      "energy": 10000
  }
}
```

The start of a charging period is a timestamp, rather than seconds since start of a transaction. This makes it easier for a CSO or EMSP to validate the price against the tariff. Start and stop time of transaction are not part of CostDetailsType, because that information is already part of the TransactionEventRequest messages. Similarly, unit prices and conditions are not part of it either, because that information is already in the TariffType referred to by *tariffId*.

Below is an example of a transaction with two ChargingPeriodType that started at 17:30h for the low price and continued after 18:00h for the higher price.

NOTE In this example the local time is in Zulu (or GMT) timezone for simplicity.

CostDetails example for tariff 11 (above) when charging 10 kWh in which 6 kWh have a high and 4 kWh have a low price

```
{
  "chargingPeriods": [{
    "tariffId": "11",
    "startPeriod": "2023-04-05T17:30:00Z",
    "dimensions": [{
      "type": "energy",
      "volume": 6000 -- 6 kWh @ 0,25
    }]
  },{
    "tariffId": "11",
    "startPeriod": "2023-04-05T18:00:00Z",
    "dimensions": [{
      "type": "energy",
      "volume": 4000 -- 4 kWh @ 0.40
    }]
  }],
  "totalCost": {
    "currency": "EUR",
    "energy": {
      "exclTax": 3.10,
      "inclTax": 3.41,
      "taxRates": [ { "type": "vat", "tax": 10.0 } ]
    },
    "total": {
      "exclTax": 3.10,
      "inclTax": 3.41
    }
  },
  "totalUsage": {
    "energy": 10000
  }
}
```

Following is an example with one ChargingPeriodType that has two CostDimensionType, because price was determined by both charging time ("ChargingTime") and maximum power ("MaxPower").

CostDetails example for tariff 12 (above) when charging 1 hour below 11 kW and half hour above 11 kW

```
{
  "chargingPeriods": [
  {
    "tariffId": "12",
    "startPeriod": "2023-04-05T14:00:00Z",
    "dimensions": [{
      "type": "ChargingTime",
      "volume": 3600
    },{
      "type": "MinPower",
      "volume": 0
    },{
      "type": "MaxPower", -- determines price level
      "volume": 10500
    }],
  },{
    "tariffId": "12",
    "startPeriod": "2023-04-05T15:00:00Z",
    "dimensions": [{
      "type": "ChargingTime",
      "volume": 3600
    },{
      "type": "MinPower", -- determines price level
      "volume": 11000
    },{
      "type": "MaxPower", -- recorded, but not relevant for price
      "volume": 22000
    }]
  }],
  "totalCost": {
    "currency": "EUR",
    "fixed": {
      "exclTax": 2.50,
      "inclTax": 2.875,
      "taxRates": [ { "type": "vat", "tax": 15.0 } ]
    },
    "chargingTime": {
      "exclTax": 2.00,
      "inclTax": 2.40,
      "taxRates": [ { "type": "vat", "tax": 20.0 } ]
    },
    "total": {
      "exclTax": 4.50,
      "inclTax": 5.275
    }
  },
  "totalUsage": {
    "chargingTime": 4800
  }
}
```

Chapter 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
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	**1.** The EV Driver wants to charge an EV, he presents his IdTokenType. **2.** The Charging Station sends AuthorizeRequest to the CSMS to request authorization. **3.** Upon receipt of AuthorizeRequest, the CSMS responds with AuthorizeResponse. This response message indicates whether or not the IdTokenType is accepted by the CSMS, and reports the EV Driver-specific tariff in the personalMessage 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)**	**Successful postcondition:** The EV Driver is authorized, knows which tariff is applicable for him/her and can start charging. **Failure postcondition:** If the authorization status is other than *Accepted*, the EV Driver can *not* start and might not know the tariff.
7	**Error Handling**	n/a
8	**Remarks** *Updated in OCPP 2.1*	This use case is only about displaying a textual representation of the tariff. This can be used when support for the TariffType structure for tariffs from I07 - Local Cost Calculation - Set Default Tariff is not supported. The tariff information presented might be the same for any token presented. If known, and applicable, it is advisable to show the tariff information in a language understood by the EV Driver. 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 at it is clear to the driver how a transaction can be canceled.

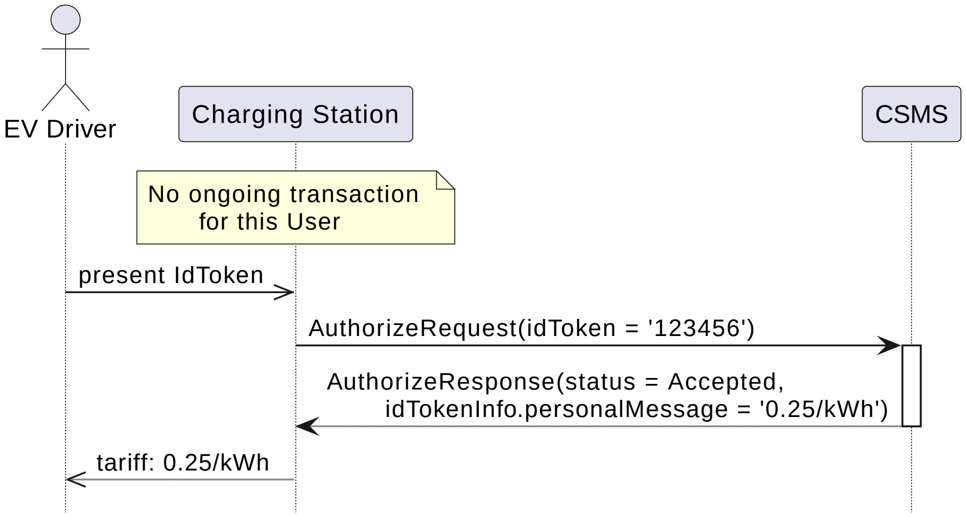

EV Driver
Charging Station
CSMS
No ongoing transaction for this User
present IdToken
AuthorizeRequest(idToken = '123456')
AuthorizeResponse(status = Accepted, idTokenInfo.personalMessage = '0.25/kWh')
tariff: 0.25/kWh

Figure 99. Sequence Diagram: Show EV Driver-specific tariff information

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
3	**Objective(s)**	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	**1.** Every Y seconds the CSMS sends a CostUpdatedRequest to the Charging Station to update the current total cost. **2.** Upon receipt of the CostUpdatedRequest, the Charging Station responds with a CostUpdatedResponse. **3.** The Charging Station shows the current total cost to the EV Driver.
	Alternative scenario	**1.** 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. **2.** 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)**	**Successful postcondition:** The EV Driver knows the running total cost during charging. **Failure postcondition:** Total cost not known to the EV Driver 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.

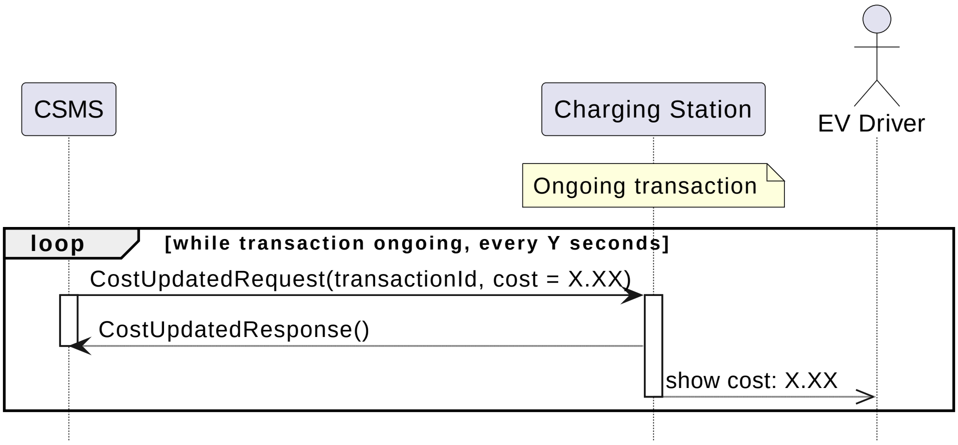

CSMS
Charging Station
EV Driver
Ongoing transaction
loop
[while transaction ongoing, every Y seconds]
CostUpdatedRequest(transactionId, cost = X.XX)
CostUpdatedResponse()
show cost: X.XX

Figure 100. Sequence Diagram: Show EV Driver Running Total Cost During Charging

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
3	**Objective(s)**	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	**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)**	**Successful postcondition:** The EV Driver knows the total cost of the transaction. **Failure postcondition:** The EV Driver does NOT know the total cost of the transaction.
7	**Error Handling**	n/a

No.	Type	Description
8	**Remarks**	If the Charging Station was offline when the transaction ended and the TransactionEventResponse with *totalCost* is received when the Charging Station comes back online some time after that, then there is no use in displaying the cost, because the driver has likely left already. A similar situation applies when `TxStopPoint` is defined as ParkingBayOccupancy, in which case the EV must leave the Charging Station to cause the transaction to end. 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 message are send. For more details see the use case: E06 - Stop Transaction options

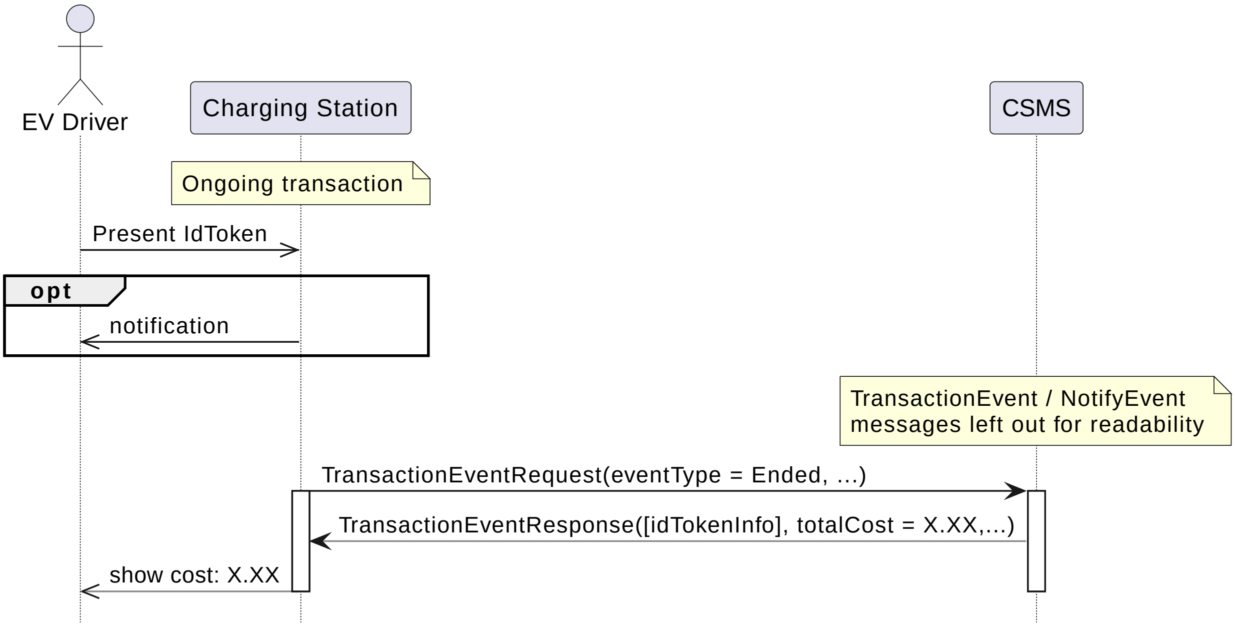

EV Driver
Charging Station
CSMS
Ongoing transaction
Present IdToken
opt
notification
TransactionEvent / NotifyEvent messages left out for readability
TransactionEventRequest(eventType = Ended, ...)
TransactionEventResponse([idTokenInfo], totalCost = X.XX,...)
show cost: X.XX

Figure 101. Sequence Diagram: Show EV Driver Final Total Cost After Charging

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 (eventType = Ended) 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 totalCost 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 *totalCost* to 0.00. Thus omitting *totalCost* does not imply that the transaction was free.
I03.FR.05	I02.FR.02 AND `TxStopPoint` 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

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	**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)**	**Successful postcondition:** EV Driver has been shown the fallback tariff information message **Failure postcondition:** EV Driver has no information about the tariff at this Charging Station.
7	**Error Handling**	n/a
8	**Remarks**	n/a

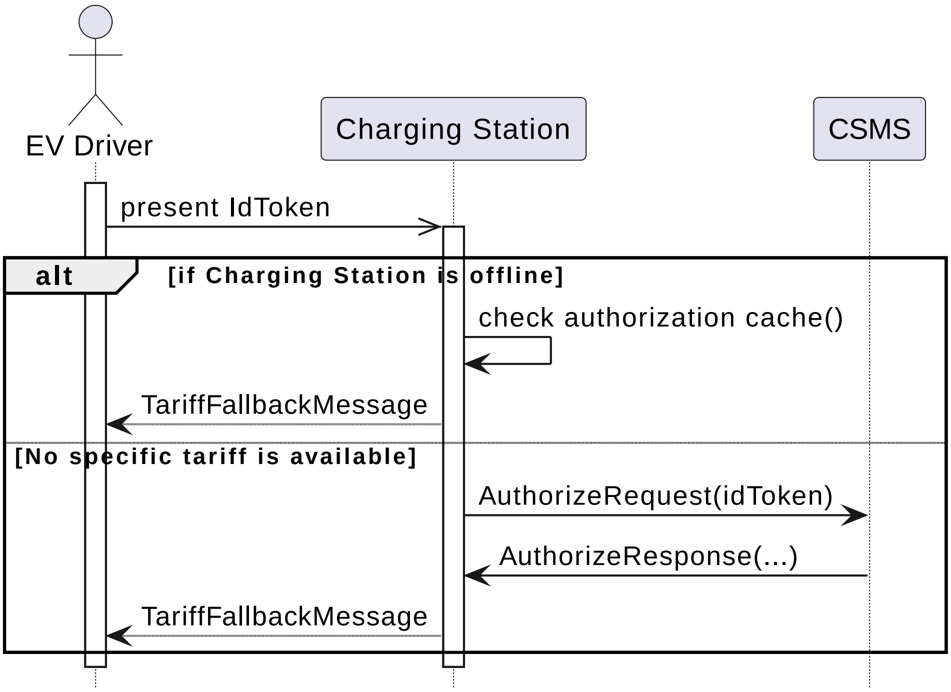

EV Driver
Charging Station
CSMS
present IdToken
alt
[if Charging Station is offline]
check authorization cache()
TariffFallbackMessage
[No specific tariff is available]
AuthorizeRequest(idToken)
AuthorizeResponse(...)
TariffFallbackMessage

Figure 102. Sequence Diagram: Show Fallback Tariff Information

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
3	**Objective(s)**	To show an EV Driver a message instead of the actual total cost when the Charging Station is *Offline* when a transaction is stopped.
4	**Description**	When an EV Driver wants to stop an ongoing transaction, but the Charging Station is *Offline*. 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	**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 *Offline* the Configuration Variable: `TotalCostFallbackMessage` is configured.
6	**Postcondition(s)**	**Successful postcondition:** The EV Driver has received a pre-configured fallback message. **Failure postcondition:** The EV Driver has not received a pre-configured fallback message.
7	**Error Handling**	n/a
8	**Remarks**	n/a

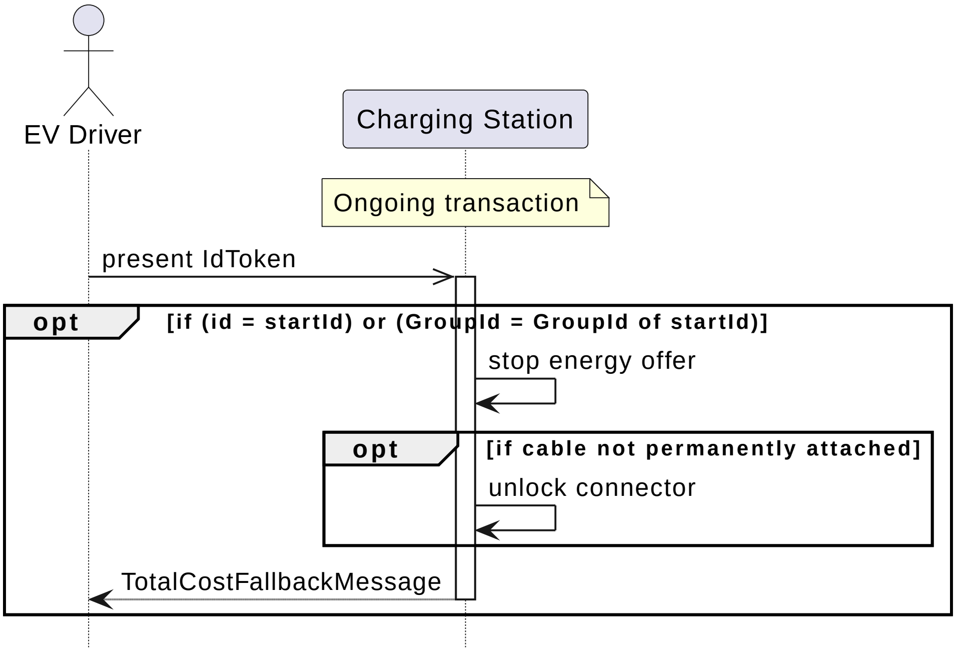


Figure 103. Sequence Diagram: Show Fallback Total Cost Message

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`.
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
3	**Objective(s)**	To show an EV Driver updated tariff information during a transaction.
4	**Description**	During charging (especially DC fast charging) it might be useful to show the EV driver updated tariff information when it becomes available. Example: If a tariff has a bandwidth: *charging will cost between 0,25 and 0,40 euro/kWh depending on current energy price. Current price is 0,28 euro/kWh.* Then when the price changing, this tariff information needs to be updated: *charging will cost between 0,25 and 0,40 euro/kWh depending on current energy price. Current price is 0,32 euro/kWh.*
	Scenario description	**1.** The Charging Station sends TransactionEventRequest (eventType = Updated) messages during the transaction. **2.** When the CSMS receives a TransactionEventRequest message it checks if there is updated tariff information available. **3.** The CSMS acknowledges with a TransactionEventResponse message, which contains the updated tariff information if available.
5	**Prerequisites**	The Charging Station supports Tariff Information There is a transaction ongoing
6	**Postcondition(s)**	**Successful postcondition:** The updated tariff information is shown to the EV Driver. **Failure postcondition:** The EV Driver has not been shown the updated tariff information.
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.

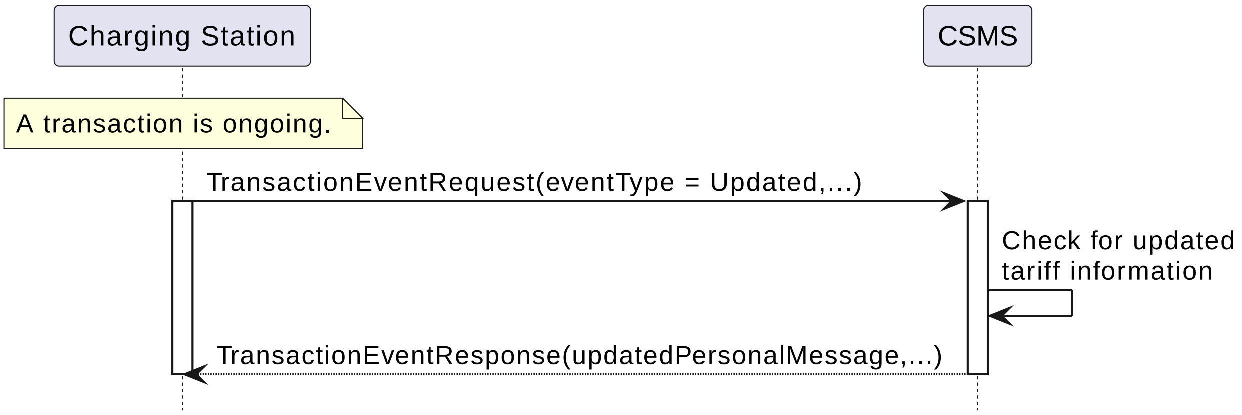

Charging Station
CSMS
A transaction is ongoing.
TransactionEventRequest(eventType = Updated,...)
Check for updated tariff information
TransactionEventResponse(updatedPersonalMessage,...)

Figure 104. Sequence Diagram: Update Tariff Information During Transaction

I06 - Update Tariff Information During Transaction - Requirements

ID.	Precondition	Requirements
I06.FR.01	When the CSMS receives a TransactionEventRequest (eventType = Updated) 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 default language in the *updatedPersonalMessage* field and optionally other languages of this in *updatedPersonalMessageExtra* fields.

ID.	Precondition	Requirements
I06.FR.03	I06.FR.02	The Charging Station SHALL display the updated tariff information to the EV Driver.

I07 - Local Cost Calculation - Set Default Tariff

New in OCPP 2.1

No.	Type	Description
1	**Name**	Local Cost Calculation - Set Default Tariff
2	**ID**	I07
3	**Objective(s)**	To set the default tariff on the charging station, for example for ad hoc charging, or when there is no driver-specific tariff.
4	**Description**	CSMS provides a tariff structure that Charging Station must use to calculate the cost. This use case assumes a Charging Station with 4 EVSEs.
	Actors	Charging Station, CSMS, EV Driver
	Scenario description	*Same tariff for all 4 EVSEs* 1. CSMS sends SetDefaultTariffRequest to Charging Station with *evseId* = 0 (to install at all EVSEs) and a *tariff* with *tariffId* = "Tariff1" that describes the tariff structure Tariff1. 2. Charging Station responds with SetDefaultTariffResponse to CSMS with *status* = `Accepted`. 3. Charging Station displays tariff description, *tariff.description* on display for all 4 EVSEs.
	Scenario description #2	*Same tariff for all EVSEs, except #4* 1. CSMS sends SetDefaultTariffRequest to Charging Station with *evseId* = 0 (to install at all EVSEs) and a *tariff* with *tariffId* = "Tariff1" that describes the Tariff1 structure for all EVSEs. 2. Charging Station responds with SetDefaultTariffResponse to CSMS with *status* = `Accepted`. 3. CSMS sends SetDefaultTariffRequest to Charging Station with *evseId* = 4 (install only at EVSE #4) and a *tariff* with *tariffId* = "Tariff2" that describes the Tariff2 structure that replaces the previously installed Tariff1. 4. Charging Station responds with SetDefaultTariffResponse to CSMS with *status* = `Accepted`. 5. Charging Station displays Tariff1 description on display for EVSE #1, #2 and #3. 6. Charging Station displays Tariff2 description on display for EVSE #4.
	Scenario description #3	*Update a default tariff* 1. CSMS sends SetDefaultTariffRequest to Charging Station for *evseId* = 0 with a *tariff* with *tariffId* = "Tariff3" and *validFrom* set to tomorrow 0:00h that describes the new tariff structure Tariff3 to become valid tomorrow. 2. Charging Station responds with SetDefaultTariffResponse to CSMS with *status* = `Accepted`. 3. At tomorrow 0:00h the new Tariff3 will become active and replace the existing default tariffs on all EVSEs.
5	**Prerequisites**	Charging Station supports local cost calculation
6	**Postcondition(s)**	
7	**Error Handling**	

No.	Type	Description
8	**Remarks**	Only one tariff structure can be sent in a message, because a tariff structure can potentially be large. Sending a SetDefaultTariffRequest with *evseId* = 0 basically copies the *tariff* to each EVSE of the Charging Station. A *tariffId* uniquely identifies the tariff for the Charging Station. An updated default tariff must therefore have a new *tariffId*. In order to update the default tariff per a certain date, a new default tariff can be installed with a *validFrom* date after which it becomes valid and replaces the existing default tariff.

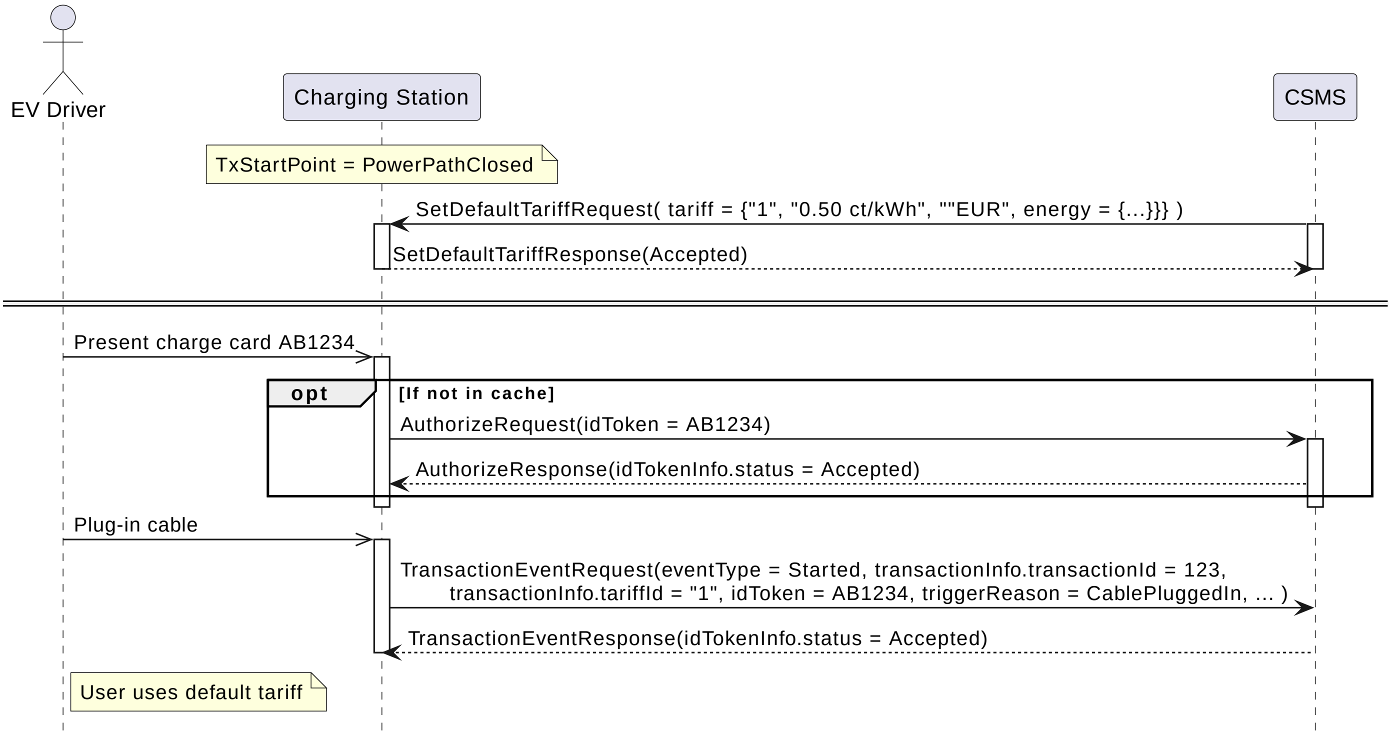

EV Driver
Charging Station
CSMS
TxStartPoint = PowerPathClosed
SetDefaultTariffRequest(tariff = {"1", "0.50 ct/kWh", ""EUR", energy = {...}}})
SetDefaultTariffResponse(Accepted)
Present charge card AB1234
opt
[If not in cache]
AuthorizeRequest(idToken = AB1234)
AuthorizeResponse(idTokenInfo.status = Accepted)
Plug-in cable
TransactionEventRequest(eventType = Started, transactionInfo.transactionId = 123, transactionInfo.tariffId = "1", idToken = AB1234, triggerReason = CablePluggedIn, ...)
TransactionEventResponse(idTokenInfo.status = Accepted)
User uses default tariff

Figure 105. Sequence diagram when using default tariff

I07 - Set Default Tariff - Requirements

ID.	Precondition	Requirements	Note
I07.FR.01	Upon receiving a SetDefaultTariffRequest when TariffCostCtrlr is absent or TariffCostCtrlr.Enabled[Tariff] = false (i.e. Charging Station does not support local cost calculation)	Charging Station SHALL respond with an RPC CALLERROR "NotSupported"	Or "NotImplemented" if the message has not been implemented.
I07.FR.02	Upon receiving a SetDefaultTariffRequest with a *tariff* structure in which the total count of TariffEnergyType, TariffTimeType and TariffFixedType elements is more than Charging Station is able to process	Charging Station SHALL respond with SetDefaultTariffResponse with *status* = `TooManyElements`	
I07.FR.03	Upon receiving a SetDefaultTariffRequest with a *tariff* structure that contains elements with a TariffConditionsType or TariffConditionsFixedType AND TariffCostCtrlr.ConditionsSupported = false (i.e. Charging Station does not support conditions)	Charging Station SHALL respond with SetDefaultTariffResponse with *status* = `ConditionNotSupported`.	
I07.FR.04	Upon receiving a SetDefaultTariffRequest with a *tariff* that has a *tariffId* that is already present in the Charging Station	Charging Station SHALL respond with SetDefaultTariffResponse with *status* = `DuplicateTariffId`.	

ID.	Precondition	Requirements	Note
I07.FR.05	Upon receiving a SetDefaultTariffRequest with an invalid *tariff* structure	Charging Station SHALL respond with SetDefaultTariffResponse with *status* = `Rejected`, optionally with a *statusInfo* element with *reasonCode* = "InvalidValue".	
I07.FR.06	Upon receiving a SetDefaultTariffRequest with an invalid *evseId*	Charging Station SHALL respond with SetDefaultTariffResponse with *status* = `Rejected`, optionally with a *statusInfo* element with *reasonCode* = "UnknownEVSE".	
I07.FR.10	Upon receiving a SetDefaultTariffRequest with a valid *tariff* structure AND NOT I07.FR.01-06	Charging Station SHALL respond with SetDefaultTariffResponse with *status* = `Accepted`	
I07.FR.11	I07.FR.10 AND Charging Station receives a SetDefaultTariffRequest with a *tariff* and *evseId* > 0	Charging Station SHALL install *tariff* as the default tariff at EVSE *evseId*.	
I07.FR.12	I07.FR.10 AND Charging Station receives SetDefaultTariffRequest with a *tariff* and *evseId* = 0	Charging Station SHALL install *tariff* as the default tariff at each EVSE.	This is exactly the same as sending a SetDefaultTariffRequest for each individual EVSE of the Charging Station.
I07.FR.13	I07.FR.10	Charging Station SHALL store the default tariff in persistent memory.	
I07.FR.14	If a default tariff has been installed for an EVSE AND No driver-specific tariff has been received for the *idToken* for which a transaction has started or will be started.	Charging Station SHALL show and use the default tariff for cost calculation.	A default tariff may also be communicated through other means than the Charging Station display.
I07.FR.17		A *startTimeOfDay*, *endTimeOfDay*, *validFromDate* and *validToDate* in a TariffConditionsType or TariffConditionsFixedType SHALL be in local time.	This is a five character time notation in "hh:mm" format.
I07.FR.18		A *validFrom* in TariffType SHALL be specified in system time.	This is usually in UTC ("Z" time zone).
I07.FR.19	If TariffType has a value for *validFrom*	Charging Station SHALL NOT consider this tariff for use until the current time is equal or later than *validFrom*.	
I07.FR.20	If a tariff A is present on EVSE X with no *validFrom* AND a tariff B is present on EVSE X with a value for *validFrom* AND current time is equal or later than B.*validFrom* (i.e. B is now valid)	Charging Station SHALL replace the tariff A on EVSE X with the new tariff B.	
I07.FR.21	If a tariff A is present on EVSE X with A.*validFrom* = T1 AND a tariff B is present on EVSE X with a value for B.*validFrom* = T2 AND current time is equal or later than B.*validFrom* (i.e. B is now valid) AND A.*validFrom* <= B.*validFrom*	Charging Station SHALL replace the tariff A on EVSE X with the new tariff B.	
I07.FR.22	If a tariff A is present on EVSE X with no *validFrom* AND a tariff B is installed on EVSE X with no *validFrom*	Charging Station SHALL replace the tariff A on EVSE X with the new tariff B.	

ID.	Precondition	Requirements	Note
I07.FR.23	If a tariff A is present on EVSE X with A.*validFrom* = T1 AND a tariff B is present on EVSE X with a value for B.*validFrom* = T2 AND current time is less than A.*validFrom* (i.e. A is not yet valid) AND A.*validFrom* > B.*validFrom*	Charging Station SHALL NOT replace the tariff A on EVSE X with the new tariff B.	A becomes valid after B, so must not be removed.
I07.FR.24	(I07.FR.20 OR I07.FR.21 OR I07.FR.22) AND the tariff A is used in an active transaction on EVSE X	Charging Station SHALL continue to use the tariff A for the transaction on EVSE X until the transaction has ended.	The tariff cannot change during a transaction, unless explicitly requested via ChangeTransactionTariffRequest.
	Tariff acceptance		
I07.FR.30	I07.FR.14 AND EV Driver does not accept the tariff AND Charging Station has already started a transaction	Charging Station SHALL deauthorize the *idToken* of the transaction and send a TransactionEventRequest with *triggerReason* = "TariffNotAccepted".	There will be no *tariffId* and no cost associated with transaction. Without authorization, no energy will be delivered. If TxStopPoint contains "Authorized" or "PowerPathClosed" this situation will end the transaction with *stoppedReason* = `DeAuthorized`.
I07.FR.31	I07.FR.14 AND EV Driver does not accept the tariff AND Charging Station has not yet started a transaction	Charging Station SHALL deauthorize the *idToken* of EV Driver.	A transaction can still be started after this when a cable is plugged in and TxStartPoint = "EVConnected", but without authorization, no energy will be delivered.
I07.FR.32	I07.FR.14 AND EV Driver has implicitly or explicitly accepted the tariff	Charging Station SHALL associate the default tariff with the transaction by including *transactionInfo.tariffId* once in the next TransactionEventRequest message.	

NOTE A "default tariff" is a tariff installed via a SetDefaultTariffRequest message.

I08 - Local Cost Calculation - Receive Driver Tariff

New in OCPP 2.1

No.	Type	Description
1	**Name**	Local Cost Calculation - Receive Driver Tariff
2	**ID**	I08
3	**Objective(s)**	To receive the driver-specific tariff in response to an AuthorizeRequest.
4	**Description**	CSMS provides a tariff structure in a AuthorizeResponse that Charging Station must use to calculate the cost. This implies that Authorization Cache or Local Authorization List cannot be used.
	Actors	Charging Station, CSMS, EV Driver

No.	Type	Description
	Scenario description #1	*Driver authorizes first (TxStartPoint = PowerPathClosed)* 1. Driver presents *idToken* for authorization. a. Charging Station sends an AuthorizeRequest for *idToken*. b. CSMS determines that a driver-specific tariff exists for this *idToken* and responds with an AuthorizeResponse with *idTokenInfo* which has *status* = `Accepted` and the tariff for the driver in *tariff*. 2. Charging Station shows the tariff to driver. **NOTE** If driver had already plugged-in, then charging would immediately start after authorization. In order to give the driver time to read and accept the tariff, Charging Station may decide to delay the charging until this has happened. 3. Driver connects the charging cable and accepts the tariff. 4. Charging Station starts a transaction and associates it with the driver-specific tariff. a. Charging Station sends a TransactionEventRequest with *eventType* = `Started`, *triggerReason* = `CablePluggedIn`, the *idToken*, the *evse* that is used, *chargingState* = `Charging` and *transactionInfo.tariffId* set to the driver-specific tariff (plus any other relevant fields). b. CSMS responds with a TransactionEventResponse with *idTokenInfo.status* = `Accepted`.
	Scenario description #2	*Driver plugs-in first (TxStartPoint = EVConnected)* 1. Driver connects the charging cable. a. Charging Station sends a TransactionEventRequest with *eventType* = `Started`, *triggerReason* = `CablePluggedIn`, the *evse* that is used, *chargingState* = `EVConnected` (and any other relevant fields). b. CSMS responds with a TransactionEventResponse. 2. Driver presents *idToken* for authorization. a. Charging Station sends an AuthorizeRequest for *idToken*. b. CSMS determines that a driver-specific tariff exists for this *idToken* and responds with an AuthorizeResponse with *idTokenInfo* which has *status* = `Accepted` and the tariff for the driver in *tariff*. c. Charging Station sends a TransactionEventRequest with *eventType* = `Updated`, *triggerReason* = `Authorized`, *chargingState* = `EVConnected` (and any other relevant fields). d. CSMS responds with a TransactionEventResponse with *idTokenInfo.status* = `Accepted`. 3. Charging Station delays the charging until driver accepts tariff. 4. Charging Station shows the tariff to driver. 5. Driver accepts tariff. 6. Charging Station associates the transaction with the tariff. a. Charging Station sends a TransactionEventRequest with *eventType* = `Updated`, *triggerReason* = `ChargingStateChanged`, *chargingState* = `Charging` and *transactionInfo.tariffId* set to the driver-specific tariff (plus any other relevant fields). b. CSMS responds with a TransactionEventResponse.
5	**Prerequisites**	Authorization Cache or Local Authorization List are not enabled. Charging Station supports local cost calculation.
6	**Postcondition(s)**	Charging Station does local cost calculation for transaction.
7	**Error Handling**	A tariff structure in field *tariff* of AuthorizeResponse can be complex. If Charging Station fails to parse *tariff* (for whatever reason), it will respond with an RPC Framework CALLRESULTERROR(FormatViolation or InternalError). Charging Station will not allow *idToken* to start charging.

No.	Type	Description
8	**Remarks**	A tariff is returned for the *idToken* at a point in time where the EVSE to be used may not yet be known. In the situation where a Charging Station has a low power and a high power EVSE with different pricing, then a tariff with restrictions on min/max power or EVSE type AC/DC needs to be used to differentiate the pricing. How a driver accepts the tariff is up to user interface of the Charging Station. If the Charging Station does not have any method to accept user input (e.g. no buttons), then a work-around can be to request the driver to present the idToken once more to confirm.

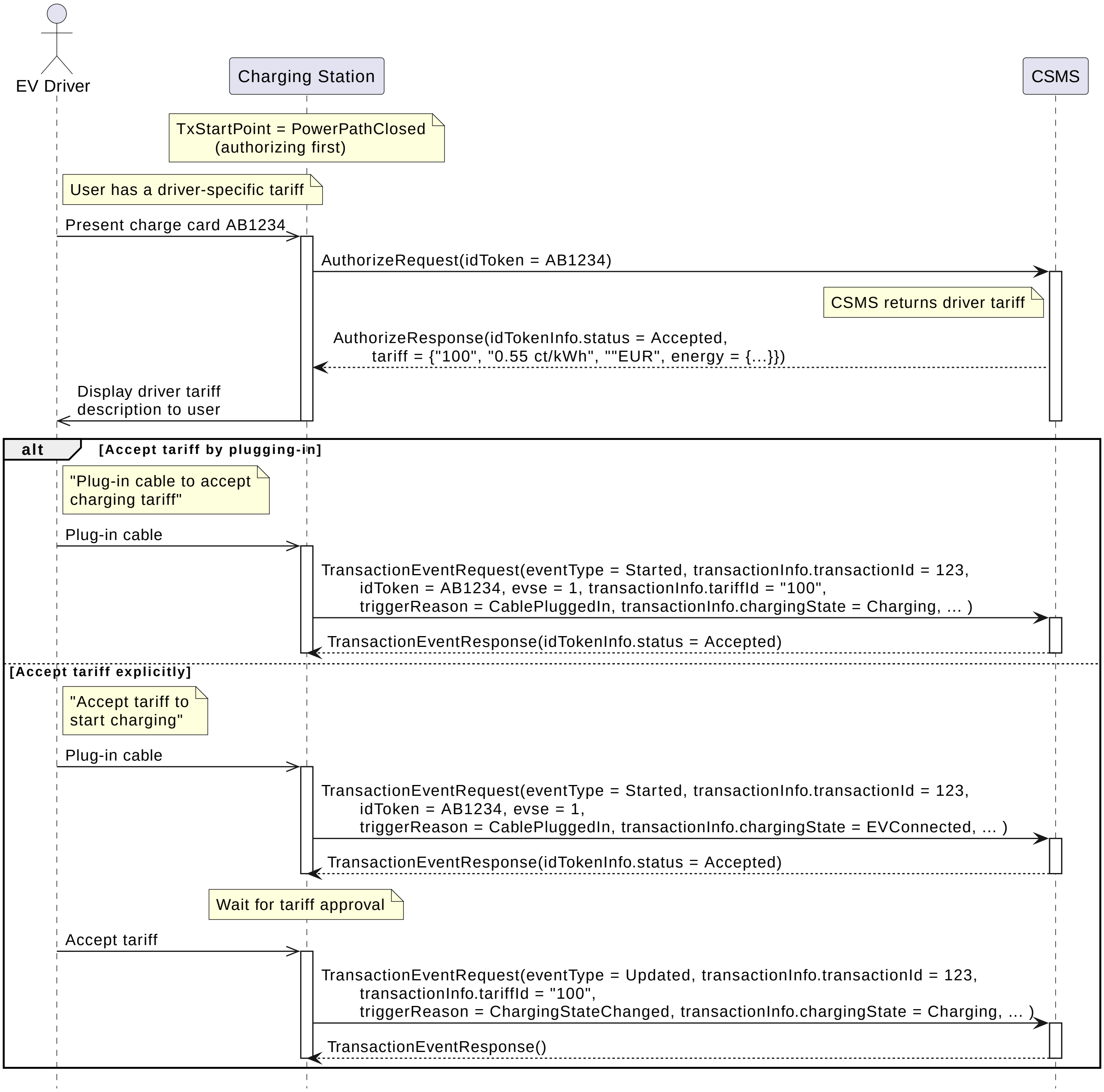

EV Driver
Charging Station
CSMS
TxStartPoint = PowerPathClosed
(authorizing first)
User has a driver-specific tariff
Present charge card AB1234
AuthorizeRequest(idToken = AB1234)
CSMS returns driver tariff
AuthorizeResponse(idTokenInfo.status = Accepted,
tariff = {"100", "0.55 ct/kWh", ""EUR", energy = {...}})
Display driver tariff
description to user
alt
[Accept tariff by plugging-in]
"Plug-in cable to accept
charging tariff"
Plug-in cable
TransactionEventRequest(eventType = Started, transactionInfo.transactionId = 123,
idToken = AB1234, evse = 1, transactionInfo.tariffId = "100",
triggerReason = CablePluggedIn, transactionInfo.chargingState = Charging, ...)
TransactionEventResponse(idTokenInfo.status = Accepted)
[Accept tariff explicitly]
"Accept tariff to
start charging"
Plug-in cable
TransactionEventRequest(eventType = Started, transactionInfo.transactionId = 123,
idToken = AB1234, evse = 1,
triggerReason = CablePluggedIn, transactionInfo.chargingState = EVConnected, ...)
TransactionEventResponse(idTokenInfo.status = Accepted)
Wait for tariff approval
Accept tariff
TransactionEventRequest(eventType = Updated, transactionInfo.transactionId = 123,
transactionInfo.tariffId = "100",
triggerReason = ChargingStateChanged, transactionInfo.chargingState = Charging, ...)
TransactionEventResponse()

Figure 106. Sequence diagram TxStartPoint = PowerPathClosed, authorize first

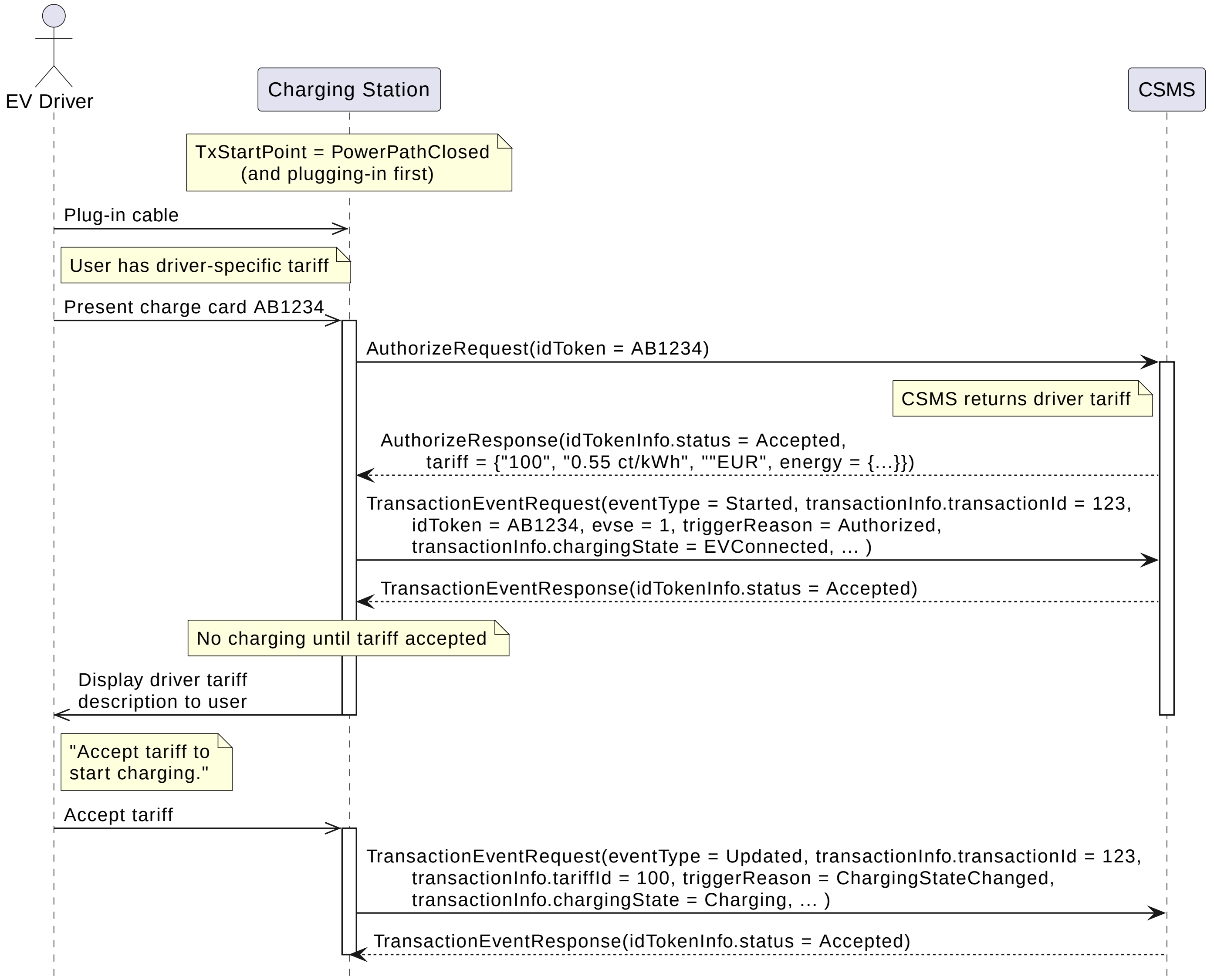

EV Driver
Charging Station
CSMS
TxStartPoint = PowerPathClosed
(and plugging-in first)
Plug-in cable
User has driver-specific tariff
Present charge card AB1234
AuthorizeRequest(idToken = AB1234)
CSMS returns driver tariff
AuthorizeResponse(idTokenInfo.status = Accepted,
tariff = {"100", "0.55 ct/kWh", ""EUR", energy = {...}})
TransactionEventRequest(eventType = Started, transactionInfo.transactionId = 123,
idToken = AB1234, evse = 1, triggerReason = Authorized,
transactionInfo.chargingState = EVConnected, ...)
TransactionEventResponse(idTokenInfo.status = Accepted)
No charging until tariff accepted
Display driver tariff
description to user
"Accept tariff to
start charging."
Accept tariff
TransactionEventRequest(eventType = Updated, transactionInfo.transactionId = 123,
transactionInfo.tariffId = 100, triggerReason = ChargingStateChanged,
transactionInfo.chargingState = Charging, ...)
TransactionEventResponse(idTokenInfo.status = Accepted)

Figure 107. Sequence diagram TxStartPoint = PowerPathClosed, plug-in first

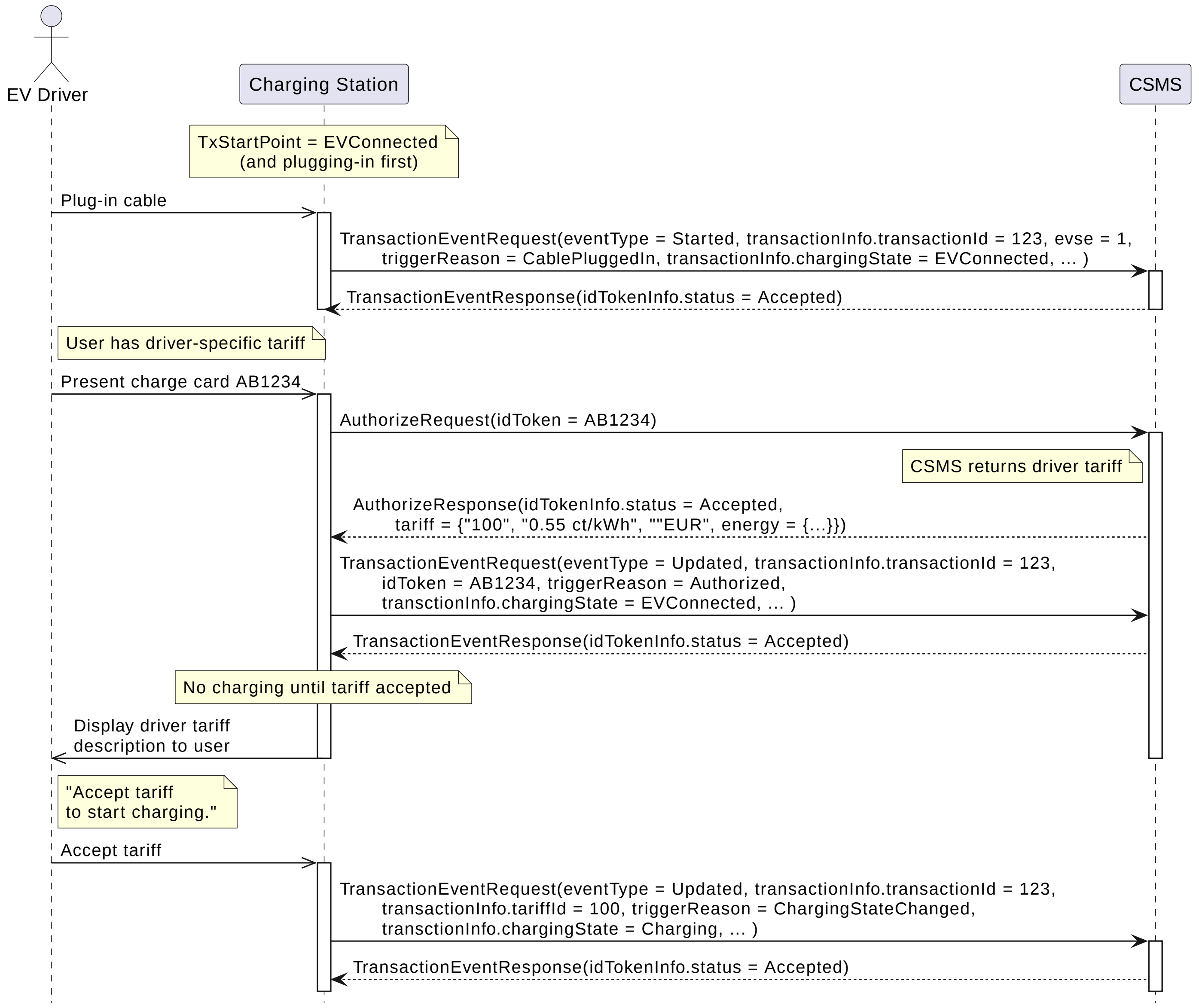

EV Driver
Charging Station
CSMS
TxStartPoint = EVConnected (and plugging-in first)
Plug-in cable
TransactionEventRequest(eventType = Started, transactionInfo.transactionId = 123, evse = 1, triggerReason = CablePluggedIn, transactionInfo.chargingState = EVConnected, ...)
TransactionEventResponse(idTokenInfo.status = Accepted)
User has driver-specific tariff
Present charge card AB1234
AuthorizeRequest(idToken = AB1234)
CSMS returns driver tariff
AuthorizeResponse(idTokenInfo.status = Accepted, tariff = {"100", "0.55 ct/kWh", ""EUR", energy = {...}})
TransactionEventRequest(eventType = Updated, transactionInfo.transactionId = 123, idToken = AB1234, triggerReason = Authorized, transctionInfo.chargingState = EVConnected, ...)
TransactionEventResponse(idTokenInfo.status = Accepted)
No charging until tariff accepted
Display driver tariff description to user
"Accept tariff to start charging."
Accept tariff
TransactionEventRequest(eventType = Updated, transactionInfo.transactionId = 123, transactionInfo.tariffId = 100, triggerReason = ChargingStateChanged, transctionInfo.chargingState = Charging, ...)
TransactionEventResponse(idTokenInfo.status = Accepted)

Figure 108. Sequence diagram TxStartPoint = EVConnected, plug-in first

I08 - Receive Driver Tariff - Requirements

ID.	Precondition	Requirements	Notes
I08.FR.01	When CSMS receives an AuthorizeRequest for an *idToken* that is accepted for charging and for which a driver-specific tariff exists	CSMS SHALL respond with an AuthorizeResponse with a *tariff* field and any other relevant fields (like *idTokenInfo*).	The details of the authorization process are described in chapter C. Authorization.
I08.FR.04	When Charging Station receives an AuthorizeResponse in which *idTokenInfo.status* = `Accepted` and with a *tariff* field which it is able to process	Charging Station SHALL use this driver-specific tariff only for the duration of the transaction that has started or will be started for *idToken*.	Depending on TxStartPoint and order of events the transaction may have started already.
I08.FR.05	I08.FR.04 AND `AuthCacheEnabled` = true	Charging Station SHALL store the driver-specific tariff for *idToken* in the Authorization Cache.	Be aware that as long as *idToken* is valid in Authorization Cache, no authorization and therefore no new tariff will be requested for the EV Driver.

ID.	Precondition	Requirements	Notes
I08.FR.06	After a transaction has ended, that has an associated driver-specific tariff in *transactionInfo.tariffId* AND This driver-specific tariff is no longer in use at Charging Station	Charging Station SHALL remove the associated tariff with *tariffId* = *transactionInfo.tariffId*.	If the tariff is also in use at another EVSE it cannot be removed.
I08.FR.07		A driver-specific tariff is considered to be "in use" at a Charging Station after an AuthorizeResponse with a tariff has been received, and until an EVConnectionTimeout occurs or the transaction that is started for the *idToken* has finished.	
I08.FR.08		CSMS SHALL ensure that TariffTypes with a different content have a different *tariffId*, such that the *tariffIds* in CostDetailsType records from historical transactions remain valid.	See use case I12 about reporting cost details at end of transaction. Choice of *tariffId* is up to CSMS. It can, for example, be a sequential number, a UUID or a hash value of tariff contents.
I08.FR.09	I08.FR.01	CSMS SHALL NOT provide the field *validFrom* in a TariffType in AuthorizeResponse.	Driver-specific tariffs have no *validFrom* date.
I08.FR.10	I08.FR.04 AND TariffType in AuthorizeResponse has a value for *validFrom*	ChargingStation SHALL ignore the value of *validFrom*.	
		Tariff acceptance	
I08.FR.20	I08.FR.04 AND EV Driver does not accept the tariff AND Charging Station has already started a transaction	Charging Station SHALL deauthorize the *idToken* of the transaction and send a TransactionEventRequest with *triggerReason* = "TariffNotAccepted".	There will be no *tariffId* and no cost associated with transaction. Without authorization, no energy will be delivered. If TxStopPoint contains "Authorized" or "PowerPathClosed" this situation will end the transaction with *stoppedReason* = `DeAuthorized`. (Same as I07.FR.30)
I08.FR.21	08.FR.04 AND EV Driver does not accept the tariff AND Charging Station has not yet started a transaction	Charging Station SHALL deauthorize the *idToken* of EV Driver.	A transaction can still be started, for example, with TxStartPoint = "EVConnected" upon cable plug-in, but without authorization, no energy will be delivered. (Same as I07.FR.31)
I08.FR.22	I08.FR.04 AND EV Driver has implicitly or explicitly accepted the tariff	Charging Station SHALL associate the tariff with the transaction by including *transactionInfo.tariffId* once in the next TransactionEventRequest message.	Implicit acceptance of tariff can, for example, be plugging cable into EV. (Same as I07.FR.32)
		Failure to process driver-specific tariff	

ID.	Precondition	Requirements	Notes
I08.FR.30	When Charging Station receives an AuthorizeResponse in which *idTokenInfo.status* = `Accepted`, but with a *tariff* field which it cannot process	Charging Station SHALL set device model variable TariffCostCtrlr.Problem to "true"	Set the Problem indicator for TariffCostCtrlr when tariff could not be processed.
I08.FR.31	I08.FR.30 AND Charging Station does not have a Delta monitor installed on TariffCostCtrlr.Problem	Charging Station SHALL send a NotifyEventRequest with *trigger* = `Alerting`, *eventNotificationType* = `HardWiredNotification`, *component* = "TariffCostCtrlr", *variable* = "Problem", *actualValue* = "true" and *techCode* optionally set to the applicable reason code from Appendix 5, to notify CSMS that it cannot support the *tariff* in the response.	*techCode* can be, for example, one of "TooManyElements", "OutOfMemory", "InternalError", "UnsupportedParam", etc.
I08.FR.31	I08.FR.30 AND `HandleFailedTariff` = "Deauthorize"	Charging Station SHALL NOT authorize *idToken* for charging.	Depending on TxStartPoint a transaction may already have started, but no energy will be delivered.
I08.FR.32	I08.FR.30 AND `HandleFailedTariff` = "UseDefault"	Charging Station SHALL authorize *idToken* for charging and SHALL use the default tariff	This is the same as if no *tariff* had been returned in AuthorizeResponse.
I08.FR.33	I08.FR.30 AND `HandleFailedTariff` = "CentralCost"	Charging Station SHALL authorize *idToken* for charging and SHALL NOT do local cost calculation	Cost will be calculated by CSMS.
I08.FR.34	I08.FR.33	CSMS SHALL do cost calculation for the transaction, as described in use cases I02 - Show EV Driver Running Total Cost During Charging and I03 - Show EV Driver Final Total Cost After Charging.	Cost will be calculated by CSMS.
I08.FR.35	I08.FR.04 AND the device model variable TariffCostCtrlr.Problem = "true"	Charging Station SHALL set device model variable TariffCostCtrlr.Problem to "false"	Remove previous Problem indicator for TariffCostCtrlr when tariff successfully received.
I08.FR.36	I08.FR.35 AND Charging Station does not have a Delta monitor installed on TariffCostCtrlr.Problem	Charging Station SHALL send a NotifyEventRequest with *trigger* = `Alerting`, *eventNotificationType* = `HardWiredNotification`, *component* = "TariffCostCtrlr", *variable* = "Problem", *actualValue* = "false"	

NOTE A "driver-specific tariff" is a tariff that was installed via an AuthorizeResponse or ChangeTransactionTariffRequest message.

I09 - Local Cost Calculation - Get Tariffs

New in OCPP 2.1

No.	Type	Description
1	**Name**	Local Cost Calculation - Get Tariffs
2	**ID**	I09
3	**Objective(s)**	To get a list of tariffs (both default and driver-specific) currently in the Charging Station.
4	**Description**	CSMS requests a list of tariffs from Charging Station. The response lists per EVSE which default tariff is installed. (A default tariff that was set on EVSE #0 is installed on every EVSE of the Charging Station). Driver-specific tariffs are reported with an idToken (after authorization) and also with an EVSE if already associated with a transaction.
	Actors	Charging Station, CSMS, EV Driver

No.	Type	Description
	Scenario description	*Charging Station has 2 EVSEs* 1. CSMS sends SetDefaultTariffRequest for *tariffId* = "Default01" with *validFrom* = "2024-01-01T00:00:00Z" to *evseId* = 0 (all EVSEs). a. Charging Station accepts with SetDefaultTariffResponse. 2. Charging Station sends AuthorizeRequest for *idToken* = "ABCD1234". a. CSMS responds with AuthorizeResponse with *tariffId* = "MSP01" and authorizes *idToken*. b. Charging Stations starts a transaction for *idToken* on EVSE #1. 3. Charging Station sends AuthorizeRequest for *idToken* = "FBFB0000". a. CSMS responds with AuthorizeResponse with *tariffId* = "MSP02" and authorizes *idToken*. 4. CSMS sends GetTariffsRequest with *evseId* = 0 to Charging Station before a transaction is started on EVSE #2 and before and EVConnectionTimeout has occurred. 5. Charging Station returns a GetTariffsResponse with all default tariffs and driver-specific tariffs that are present on the Charging Station in an array of *tariffAssignments*: ◦ *tariffId* = "Default01", *tariffKind* = `DefaultTariff`, *validFrom* = "2024-01-01T00:00:00Z", *evseIds* = [1, 2] ◦ *tariffId* = "MSP01", *tariffKind* = `DriverTariff`, *evseIds* = [1], *idTokens* = ["ABCD1234"] ◦ *tariffId* = "MSP02", *tariffKind* = `DriverTariff`, *idTokens* = ["FBFB0000"]
5	**Prerequisites**	Local cost calculation is supported.
6	**Postcondition(s)**	
7	**Error Handling**	
8	**Remarks**	

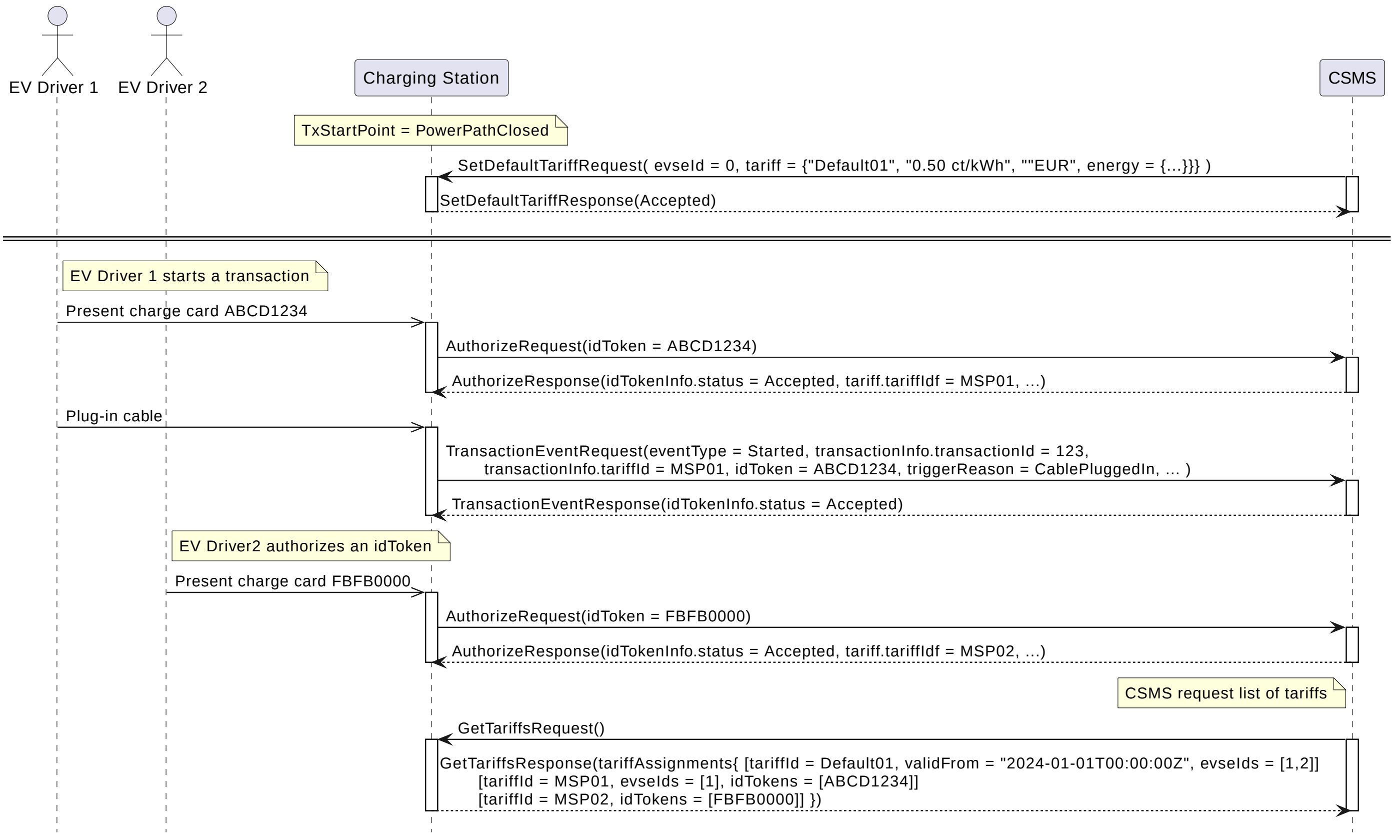

EV Driver 1
EV Driver 2
Charging Station
CSMS
TxStartPoint = PowerPathClosed
SetDefaultTariffRequest(evseId = 0, tariff = {"Default01", "0.50 ct/kWh", ""EUR", energy = {...}}})
SetDefaultTariffResponse(Accepted)
EV Driver 1 starts a transaction
Present charge card ABCD1234
AuthorizeRequest(idToken = ABCD1234)
AuthorizeResponse(idTokenInfo.status = Accepted, tariff.tariffIdf = MSP01, ...)
Plug-in cable
TransactionEventRequest(eventType = Started, transactionInfo.transactionId = 123,
transactionInfo.tariffId = MSP01, idToken = ABCD1234, triggerReason = CablePluggedIn, ...)
TransactionEventResponse(idTokenInfo.status = Accepted)
EV Driver2 authorizes an idToken
Present charge card FBFB0000
AuthorizeRequest(idToken = FBFB0000)
AuthorizeResponse(idTokenInfo.status = Accepted, tariff.tariffIdf = MSP02, ...)
CSMS request list of tariffs
GetTariffsRequest()
GetTariffsResponse(tariffAssignments{ [tariffId = Default01, validFrom = "2024-01-01T00:00:00Z", evseIds = [1,2]]
[tariffId = MSP01, evseIds = [1], idTokens = [ABCD1234]]
[tariffId = MSP02, idTokens = [FBFB0000]] })

Figure 109. Sequence diagram GetTariffsRequest

I09 - Get Tariffs - Requirements

ID.	Precondition	Requirements
I09.FR.01	Upon receiving a GetTariffsRequest with *evseId* = 0 and tariffs are present in Charging Station	Charging Station SHALL respond with GetTariffsResponse with *status* = `Accepted` and a list of *tariffAssigments* for all EVSEs.
I09.FR.02	Upon receiving a GetTariffsRequest with *evseId* > 0 and tariffs are present at EVSE *evseId*	Charging Station SHALL respond with GetTariffsResponse with *status* = `Accepted` and a list of *tariffAssignments* for only the tariffs present in EVSE *evseId*.
I09.FR.03	Upon receiving a GetTariffsRequest and no tariffs are present in Charging Station or provided EVSE *evseId*	Charging Station SHALL respond with GetTariffsResponse without a *tariffAssignments* field and with a *status* = `NoTariff`
I09.FR.04	If GetTariffsResponse contains tariffs with *tariffKind* = `DefaultTariff`	*tariffAssignments* in the GetTariffsResponse SHALL contain for each *tariffId* the list of *evseIds* where this tariff is installed.
I09.FR.05	If GetTariffsResponse contains tariffs with *tariffKind* = `DriverTariff`	*tariffAssignments* in the GetTariffsResponse SHALL contain for each *tariffId* the list of *evseIds* where this tariff is installed.
I09.FR.06	I09.FR.05 AND An active transaction is associated with the tariff, i.e. transaction has *transactionInfo.tariffId* = *tariffId*	*tariffAssignments* element SHALL also contain the EVSE where transaction is active in *evseIds* .
I09.FR.07	If GetTariffsResponse contains tariffs that have a value for *validFrom*	The *tariffAssignments* element in GetTariffsResponse SHALL contain *validFrom* for these tariffs.

I10 - Local Cost Calculation - Clear Tariffs

New in OCPP 2.1

No.	Type	Description
1	**Name**	Local Cost Calculation - Clear Tariffs
2	**ID**	I10
3	**Objective(s)**	To clear (remove) one or more default tariffs from a Charging Station
4	**Description**	CSMS requests to clear one or more default tariffs from a Charging Station. ClearTariffsRequest only applies to default tariffs. Driver-specific tariffs are automatically removed when no longer in use. A default tariff that is currently in use can be cleared, but remains valid where it is used.
	Actors	Charging Station, CSMS
	Scenario description	*Charging Station has 2 EVSEs* 1. A transaction is active on EVSE #1 with default tariff "Default01" and a transaction with a driver-specific tariff for *idToken* "FBFB0000" is active on EVSE #2. 2. The following tariffs are present, as would be shown by a GetTariffsResponse: ◦ *tariffId* = "Default01", *tariffKind* = `DefaultTariff`, *evseIds* = [1, 2] ◦ *tariffId* = "MSP01", *tariffKind* = `DriverTariff`, *idTokens* = ["FBFB0000"], *evseIds* = [2] 3. CSMS removes default tariffs "Default01" via ClearTariffsRequest with *tariffIds* = ["Default01"]. a. Charging Station clears "Default01" from both EVSE #1 and EVSE #2, but the tariff remains in use for the currently active transaction on EVSE #1. i. The driver-specific tariff "MSP01" in use on EVSE #2 is not affected. b. Charging Station responds with ClearTariffsResponse with 1 *clearTariffResult* item with *status* = `Accepted` for *tariffId* "Default01".
5	**Prerequisites**	Local cost calculation is supported.
6	**Postcondition(s)**	One or more tariffs are cleared from Charging Station, but running transactions are not affected.
7	**Error Handling**	

No.	Type	Description
8	**Remarks**	

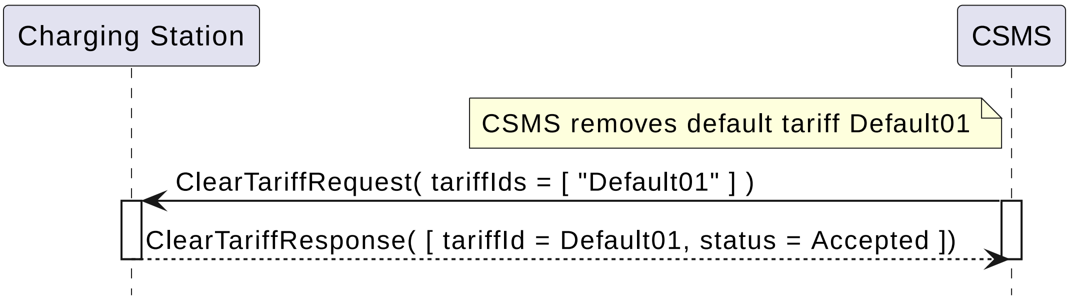

Charging Station
CSMS
CSMS removes default tariff Default01
ClearTariffRequest(tariffIds = ["Default01"])
ClearTariffResponse([tariffId = Default01, status = Accepted])

Figure 110. Sequence diagram ClearTariffsRequest

I10 - Clear Tariffs - Requirements

ID.	Precondition	Requirements
I10.FR.01	Upon receiving a ClearTariffsRequest without parameters and tariffs are present in Charging Station	Charging Station SHALL clear all tariffs in Charging Station and respond with ClearTariffsResponse with a list of *clearTariffResult* for each cleared *tariffId* and *status* = `Accepted`.
I10.FR.02	Upon receiving a ClearTariffsRequest with the parameter *tariffIds*	Charging Station SHALL clear all tariffs in Charging Station that have a *tariffId* that is present in the parameter *tariffIds*, and respond with ClearTariffsResponse with a list of cleared *tariffIds* and *status* = `Accepted`.
I10.FR.03	Upon receiving a ClearTariffsRequest with the parameter *evseId*	Charging Station SHALL clear all tariffs in Charging Station that have been set for EVSE *evseId*, and respond with ClearTariffsResponse with a list of cleared *tariffIds* and *status* = `Accepted`.
I10.FR.05	Upon receiving a ClearTariffsRequest with both parameters *tariffIds* and *evseId*	Charging Station SHALL only clear tariffs that match both parameters, and respond with ClearTariffsResponse with a list of cleared *tariffIds* and *status* = `Accepted`.
I10.FR.06	Upon receiving a ClearTariffsRequest AND no tariffs exist, or no tarifss exist that match the parameters *tariffIds* and/or *evseId*.	Charging Station SHALL respond with ClearTariffsResponse without *tariffIds* and with *status* = `NoTariff`.
I10.FR.07	Upon receiving a ClearTariffsRequest that affects a tariff is associated with a transaction, i.e. transaction has *transactionInfo.tariffId* = *tariffId*	Charging Station SHALL clear this tariff and report this tariff in *clearTariffsResult* with *tariffId* and *status* = `Accepted`, but SHALL continue to use this tariff in the transaction until it ends.

I11 - Local Cost Calculation - Change transaction tariff

New in OCPP 2.1

No.	Type	Description
1	**Name**	Local Cost Calculation - Change transaction tariff
2	**ID**	I11
3	**Objective(s)**	Change the tariff to use during a transaction
4	**Description**	CSMS changes the tariff that is associated with a transaction. This may be needed when dealing with unexpected price changes.
	Actors	Charging Station, CSMS, EV Driver

No.	Type	Description
	Scenario description	*A transaction is in progess* 1. CSMS sends a ChangeTransactionTariffRequest for *transactionId* with a new *tariff*. 2. Charging Station responds with ChangeTransactionTariffResponse with *status* = `Accepted`. 3. Charging Station starts applying the new tariff to the transaction. a. Charging Stations sends a TransactionEventRequest with *triggerReason* = `TariffChanged` and *transactionInfo.tariffId* with the *id* of the new tariff (and other relevant fields). 4. Charging Station optionally displays the new tariff that is being used. (This depends on UI design and regulations).
5	**Prerequisites**	A transaction and associated tariff are active on the Charging Station.
6	**Postcondition(s)**	Charging Station continues local cost calculation with the new tariff.
7	**Error Handling**	If Charging Station does not accept the ChangeTransactionTariffRequest then the exising tariff remains active.
8	**Remarks**	Changing a tariff during a transaction may not be allowed at all in certain legislations, or only unless specifically agreed upon by the customer. Dynamic prices based on day-ahead energy prices can already be part of a tariff with restrictions based on time of day, and do not require the tariff to be changed during a transaction. Changing of a tariff is done for a transaction and not for a driver (idToken), because the same driver could have a transaction on another EVSE as well for which no tariff update is needed.

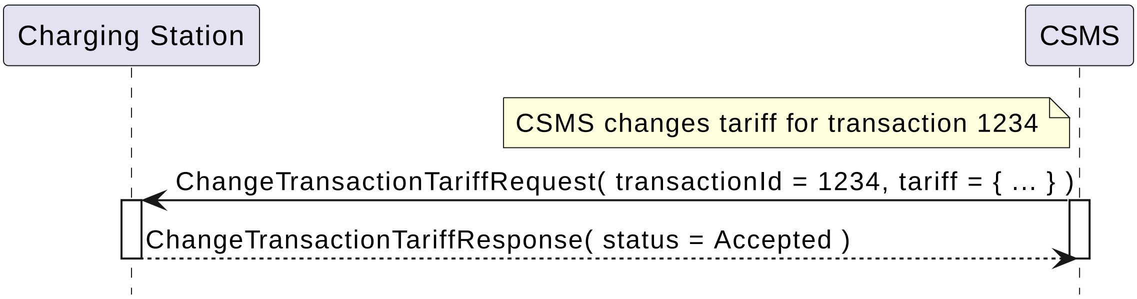

Charging Station
CSMS
CSMS changes tariff for transaction 1234
ChangeTransactionTariffRequest(transactionId = 1234, tariff = { ... })
ChangeTransactionTariffResponse(status = Accepted)

Figure 111. Sequence diagram Change TransactionTariffRequest

I11 - Change Transaction Tariff - Requirements

ID.	Precondition	Requirements	Notes
I11.FR.01	When a Charging Station receives a ChangeTransactionTariffRequest and does not support local cost calculation	Charging Station SHALL respond with an RPC CALLERROR "NotSupported" (or "NotImplemented" if the message has not been implemented).	
I11.FR.02	When a Charging Station receives a ChangeTransactionTariffRequest and *tariff* has more elements than `TariffMaxElements`	Charging Station SHALL respond with ChangeTransactionTariffResponse with *status* = `TooManyElements`.	
I11.FR.03	Upon receiving a ChangeTransactionTariffRequest with a *tariff* structure that contains elements with a TariffConditionsType or TariffConditionsFixedType AND TariffCostCtrlr.ConditionsSupported = false (i.e. Charging Station does not support conditions)	Charging Station SHALL respond with ChangeTransactionTariffResponse with *status* = `ConditionNotSupported`.	
I11.FR.04	When a Charging Station receives a ChangeTransactionTariffRequest and *transactionId* does not exist or is no longer active	Charging Station SHALL respond with ChangeTransactionTariffResponse with *status* = `TxNotFound`.	

ID.	Precondition	Requirements	Notes
I11.FR.05	When a Charging Station receives a ChangeTransactionTariffRequest and *tariff* has a value for *currency* that differs from the currently active *tariff*	Charging Station SHALL respond with ChangeTransactionTariffResponse with *status* = `NoCurrencyChange`.	It is not allowed to switch currency within a transaction.
I11.FR.06	When a Charging Station receives a ChangeTransactionTariffRequest AND NOT (I11.FR.01 to I11.FR.05)	Charging Station SHALL respond with ChangeTransactionTariffResponse with *status* = `Accepted`.	Change of tariff is accepted when none of the above error conditions apply.
I11.FR.07	I11.FR.06	Charging Station SHALL send a TransactionEventRequest with *triggerReason* = `TariffChanged` and *transactionInfo.tariffId* set to the *id* of the new tariff (and include any other relevant fields).	The new tariff is applied regardless whether a tariff was already associated with transaction or not.
I11.FR.08	I11.FR.07	Charging Station SHALL from this moment on apply the new *tariff* to the transaction.	There is no retroactive calculation of tariffs.

I12 - Local Cost Calculation - Cost Details of Transaction

New in OCPP 2.1

No.	Type	Description
1	**Name**	Local Cost Calculation - Cost Details of Transaction
2	**ID**	I12
3	**Objective(s)**	To calculate cost and provide CSMS with details about the cost
4	**Description**	Charging Station calculates cost of the transaction locally and returns a break-down of the cost at end of the transaction for every charging period for which a different tariff element was active. CSMS can use this to generate invoices or Charge Detail Records for EMSPs.
	Actors	Charging Station, CSMS, EV Driver
	Scenario description	*Cost calculation during transaction* 1. EV Driver starts charging 2. If a driver-specific tariff was provided at authorization, use the driver-specific tariff. 3. Else use the default tariff. 4. Charging Station continuously displays running cost to EV Driver. 5. While the transaction is active Charging Station provides running cost updates in TransactionEventRequest messages, depending on `RunningCostEnabled`. 6. During the transaction Charging Station may include (depending on `RunningCostEnabled`) in the TransactionEventRequest with *eventType* = `Updated` a *costDetails* field of type CostDetailsType with a breakdown of the cost of the transaction. a. *chargingPeriods* are omitted from *costDetails* that are sent during a transaction in order to limit the amount of data. b. *costDetails* includes the running total price for time, energy, fixed fee, reservation and the overall running total price of the transaction. 7. When the transaction ends Charging Station includes in the TransactionEventRequest with *eventType* = `Ended` a *costDetails* field of type CostDetailsType with a breakdown of the cost of the transaction. a. *costDetails* has a list of one or more *chargingPeriods*, where each period corresponds to a period with different pricing. b. *costDetails* also includes the total price for time, energy, fixed fee, reservation and the overall total price of the transaction.
5	**Prerequisites**	A tariff was set via SetDefaultTariffRequest or in response to an AuthorizeRequest. `TariffEnabled` = true and `CostEnabled` = true.
6	**Postcondition(s)**	CSMS received a *costDetails* with a breakdown of costs at end of transaction.
7	**Error Handling**	

No.	Type	Description
8	**Remarks**	The running cost sent with a TransactionEventRequest with *eventType* = `Updated` serves to provide details on running cost during a transaction. CSMS can, for example, communicate these values to the driver via an app. The field *costDetails* when sent as running cost does not contain any *chargingPeriods* to limit data usage during frequent updates.

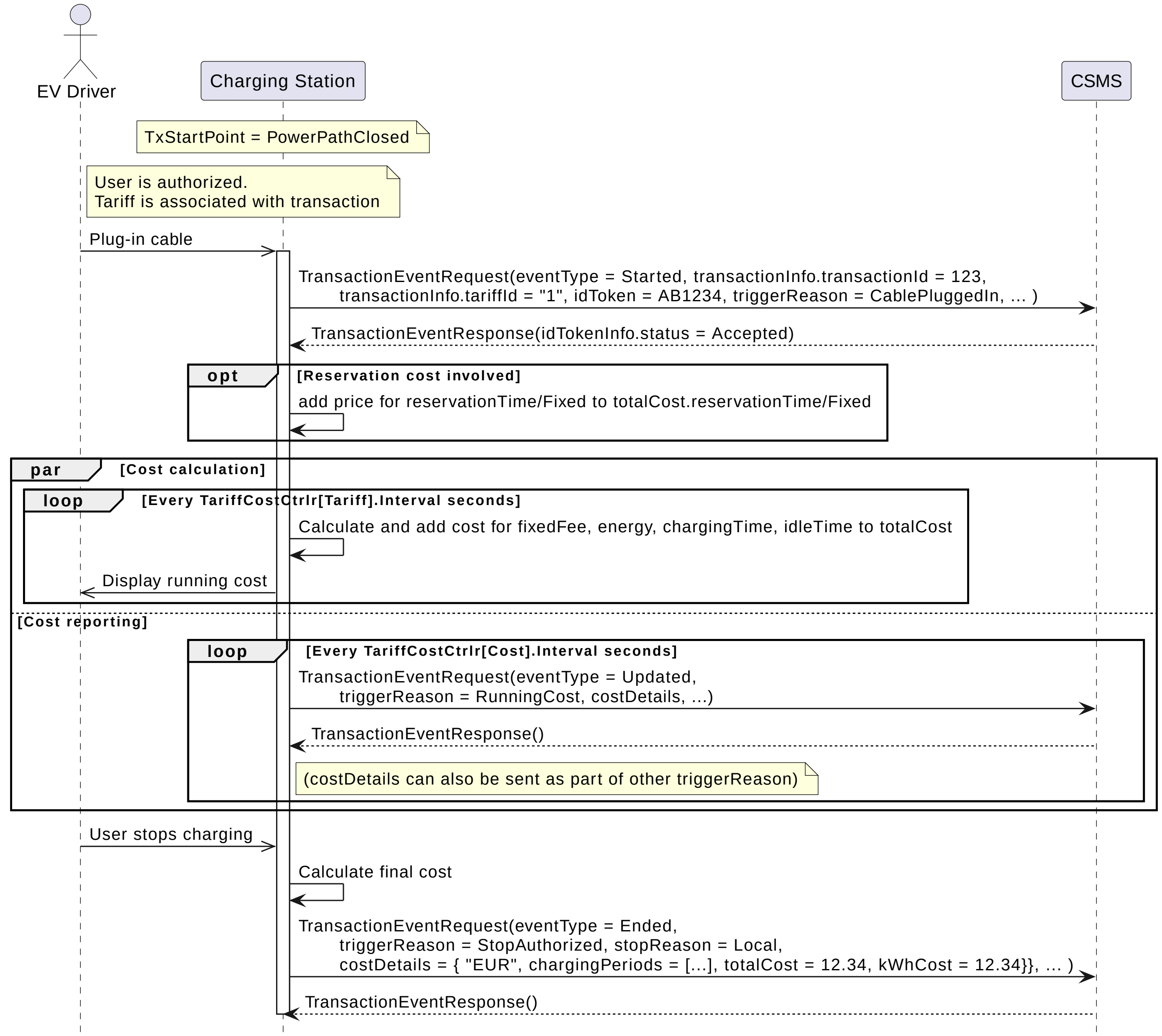

EV Driver
Charging Station
CSMS
TxStartPoint = PowerPathClosed
User is authorized.
Tariff is associated with transaction
Plug-in cable
TransactionEventRequest(eventType = Started, transactionInfo.transactionId = 123,
transactionInfo.tariffId = "1", idToken = AB1234, triggerReason = CablePluggedIn, ...)
TransactionEventResponse(idTokenInfo.status = Accepted)
opt
[Reservation cost involved]
add price for reservationTime/Fixed to totalCost.reservationTime/Fixed
par
[Cost calculation]
loop
[Every TariffCostCtrlr[Tariff].Interval seconds]
Calculate and add cost for fixedFee, energy, chargingTime, idleTime to totalCost
Display running cost
[Cost reporting]
loop
[Every TariffCostCtrlr[Cost].Interval seconds]
TransactionEventRequest(eventType = Updated,
triggerReason = RunningCost, costDetails, ...)
TransactionEventResponse()
(costDetails can also be sent as part of other triggerReason)
User stops charging
Calculate final cost
TransactionEventRequest(eventType = Ended,
triggerReason = StopAuthorized, stopReason = Local,
costDetails = { "EUR", chargingPeriods = [...], totalCost = 12.34, kWhCost = 12.34}}, ...)
TransactionEventResponse()

Figure 112. Sequence diagram for cost details

I12 - Cost Details of Transaction - Requirements

ID.	Precondition	Requirements	Notes
I12.FR.01	When a *transaction.transactionInfo.tariffId* is set AND TariffCostCtrlr.Enabled[Cost] = true AND TariffCostCtrlr.Enabled[Tariff] = true	Charging Station SHALL provide a cost breakdown in the *costDetails* field of type CostDetailsType in the final TransactionEventRequest with *eventType* = Ended of the transaction.	Tariffs and local cost calculation are supported and a tariff is associated with transaction. Tariff can be set by SetDefaultTariffRequest, AuthorizeResponse or ChangeTransactionTariffRequest.

ID.	Precondition	Requirements	Notes
I12.FR.02	I12.FR.01 AND TariffCostCtrlr.Enabled[RunningCost] = true	Charging Station SHALL provide a *costDetails* field of type CostDetailsType in TransactionEventRequest with *eventType* = `Started` and every TariffCostCtrlr.Interval[Cost] seconds during the transaction for *eventType* = `Updated`.	Providing running cost updates needs to be enabled via TariffCostCtrlr.Enabled[Running Cost]. See `RunningCostEnabled` and `CostInterval`.
I12.FR.03	I12.FR.02 AND If a TransactionEventRequest is sent only to report a running cost update	The TransactionEventRequest SHALL contain a *triggerReason* = `RunningCost`.	A running cost update is sent every `CostInterval` seconds, which can be part of a TransactionEventRequest that is sent for another *triggerReason*.
I12.FR.04	I12.FR.01	Each *chargingPeriods* element in *costDetails* SHALL contain a reference in the field *tariffId* to the tariff that was used.	
I12.FR.05	When costs are calculated from a TariffFixedPriceType	The total of these costs SHALL be in *totalCost.fixed* element	
I12.FR.06	If the transaction was authorized by an *idToken* or *groupIdToken* that was related to a reservation, as defined in Introduction	Charging Station SHALL evaluate TariffType.*reservationTime* and TariffType .*reservationFixed* elements, **once** at start of the transaction.	Reservation cost are applied once at start of transaction. See H03.FR.09.
I12.FR.07	I12.FR.06 AND Costs are calculated from TariffType .*reservationTime*	The total amount of reservation time, calculated as the duration from the time the ReserveNowRequest was received until start of the transaction, SHALL be in the *totalUsage.reservationTime* field and total of these costs SHALL be in *totalCost.reservationTime* element	This is a time-based reservation cost
I12.FR.08	I12.FR.06 AND Costs are calculated from a TariffType .*reservationFixed*	The total of these costs SHALL be in *totalCost.reservationFixed* element	This is a fixed reservation cost.
I12.FR.09	When costs are calculated from TariffEnergyPriceType	The total amount of energy SHALL be in the *totalUsage.energy* field and total of these costs SHALL be in *totalCost.energy* element	
I12.FR.10	When costs are calculated from TariffType.*chargingTime*	The total amount of charging time SHALL be in the *totalUsage.chargingTime* field and total of these costs SHALL be in *totalCost.chargingTime* element	
I12.FR.11	When costs are calculated from TariffType.*idleTime*	The total amount of idle time SHALL be in the *totalUsage.idleTime* field and total of these costs SHALL be in *totalCost.idleTime* element	
I12.FR.12		PriceType SHALL have the cost excluding tax in the *exclTax* field.	
I12.FR.13	When a PriceType has one or more associated TaxRateType	The *inclTax* field in PriceType SHALL be calculated as *exclTax* * *taxRates.tax* for the *taxRates* element with *stack* = 0 (or absent), and then the *taxRates* element(s) with *stack* = 1 is/are applied to the result for *stack* = 0, and then the element(s) *taxRates* with *stack* = 2 is/are applied to the result for *stack* = 1, etc.	For example, an "energy tax" (stack 0) is applied to the net price, and a "VAT" (stack 1) is applied on top of the net price including "energy tax". There can be multiple *taxRates* elements with the same *stack* value.
I12.FR.14	When Charging Station is not able to calculate the CostDetailsType (partly or entirely)	Charging Station SHALL set the field *failureToCalculate* to true and optionally add a *failureReason* with a human-readable text about the reason. Charging Station MAY include any values that it was able to calculate.	

ID.	Precondition	Requirements	Notes
I12.FR.15	I12.FR.14	Charging Station MAY stop sending CostDetailsType elements in TransactionEventRequest messages for the remainder of the transaction.	
I12.FR.16		The field *currency* in TotalCostType SHALL match the currency of the tariff that was used.	
I12.FR.17		The field TotalCostType.*total* SHALL contain the totals of all *exclTax* and *inclTax* parts of *fixed*, *energy*, *chargingTime*, *idleTime*, *reservationTime* and *reservationFixed* fields.	*total* is the sum of all costs.
I12.FR.18	I12.FR.14	Charging Station SHALL set device model variable TariffCostCtrlr.Problem to "true"	Set the Problem indicator for TariffCostCtrlr when tariff could not be calculated.
I12.FR.19	I12.FR.14 AND Charging Station does not have a Delta monitor installed on TariffCostCtrlr.Problem	Charging Station SHALL send a NotifyEventRequest with *trigger* = `Alerting`, *eventNotificationType* = `HardWiredNotification`, *component* = "TariffCostCtrlr", *variable* = "Problem", *actualValue* = "true"	
		Tariff evaluation	
I12.FR.30	I12.FR.01	Charging Station SHALL evaluate the *conditions* of the TariffFixedPriceType once at start of the transaction.	Fixed fee is only applied once in transaction.
I12.FR.31	I12.FR.01 AND TariffCostCtrlr.Interval[Tariff] > 0	Charging Station SHALL evaluate the *conditions* of a TariffEnergyPriceType and TariffTimePriceType for current, power and energy at least every TariffCostCtrlr.Interval[Tariff] seconds	Conditions for date and time need to be followed exactly (as these are predictable), but changes in current, power and energy only need to be monitored every `TariffInterval` seconds. The use of tariffs requires that TariffCostCtrlr.Enabled[Tariff] = true
I12.FR.32	I12.FR.01 AND TariffCostCtrlr.Interval[Tariff] = 0	Charging Station SHALL evaluate the *conditions* of a TariffEnergyPriceType, TariffTimePriceType for current, power and energy as frequent as possible.	See `TariffInterval`.
I12.FR.33	I12.FR.31 OR I12.FR.32	Charging Station SHALL create lists existing of all TariffEnergyPriceType and TariffTimePriceType for which TariffConditionsType is applicable at the moment.	This is the "applicable element list" in which multiple conditions may be applicable at this point in time.
I12.FR.34	I12.FR.33	Charging Station SHALL select for each TariffEnergyPriceType (*energy*) and TariffTimePriceType (*chargingTime*, *idleTime*) the **first** Tariff{Energy/Time}PriceType element from the "applicable element list" to contribute to the cost calculation.	This is the "used element list".
I12.FR.35	Whenever the "used element list" changes, because new TariffConditionsType apply	Charging Station SHALL end the current *chargingPeriods* entry (ChargingPeriodType) of CostDetailsType, and SHALL fill the *dimensions* fields with data of the CostDimensionType.	Based on the values in *dimensions* and the *tariffId* it is possible to verify/recalculate the cost.
I12.FR.36	I12.FR.35	Charging Station SHALL start a new *chargingPeriods* entry in CostDetailsType with *startPeriod* set to current time.	

ID.	Precondition	Requirements	Notes
I12.FR.37	When a TariffType is in use for cost calculation at the moment that local time shifts as a result of beginning or ending of daylight saving time	The execution of cost calculation SHALL follow local time for *startTimeOfDay* and *endTimeOfDay* in TariffConditionsType.	This means, for example, that a TariffEnergyPriceType that had ended at 02:30h suddenly becomes valid again half an hour later, when local time jumps back to 02:00h because daylight-saving time ends.
I12.FR.38	When TariffType contains a *minCost* AND (at the end of the transaction the total cost excluding taxes (in *costDetails*) is lower than the cost defined in *minCost.exclTax* OR at the end of the transaction the total cost including taxes is lower than the cost defined in *minCost.inclTax*)	Charging Station SHALL replace the *totalCost* element in CostDetailsType with a new TotalCostType field that has its PriceType field *fixed* set to the value of *minCost*, *typeOfCost* to `MinCost`, and *currency* set to currency of the tariff.	This replaces the calculated cost with the minimum *minCost*. *chargingPeriods* and *totalUsage* fields in CostDetailsType remain unaffected.
I12.FR.39	When TariffType contains a *maxCost* AND (during the transaction the total cost excluding taxes reaches the cost defined in *maxCost.exclTax* OR during the transaction the total cost including taxes reaches the cost defined in *maxCost.inclTax*)	Charging Station SHALL stop calculating any more costs and freeze the values in TotalCostType and set the field *typeOfCost* = `MaxCost`.	No more costs are added during the remainder of the transaction. Total amounts of prices and taxes stay at the value they had at the moment that *maxCost* was reached.
I12.FR.40	I12.FR.39	Charging Station SHALL continue to calculate usage and fill *chargingPeriods* and *totalUsage* fields in CostDetailsType	This ensures that proper usage data can be displayed on the receipt.
I12.FR.41	NOT (I12.FR.38 OR I12.FR.39)	Charging Station SHALL set *typeOfCost* in TotalCostType = `NormalCost`.	Calculation can start with *typeOfCost* = `NormalCost` and overrule by `Min/MaxCost` when applicable.
I12.FR.42	When a transaction is ending AND before Charging Station sends the TransactionEventRequest with *eventType* = `Ended`	Charging Station SHALL finish the tariff calculation and update the CostDetailsType element to be included in the TransactionEventRequest	
I12.FR.43	I12.FR.42 AND cost calculation was successful AND the device model variable TariffCostCtrlr.Problem = "true"	Charging Station SHALL set device model variable TariffCostCtrlr.Problem to "false"	Remove previous Problem indicator for TariffCostCtrlr when tariff successfully calculated.
I12.FR.44	I12.FR.43 AND Charging Station does not have a Delta monitor installed on TariffCostCtrlr.Problem	Charging Station SHALL send a NotifyEventRequest with *trigger* = `Alerting`, *eventNotificationType* = `HardWiredNotification`, *component* = "TariffCostCtrlr", *variable* = "Problem", *actualValue* = "false"	

J. Meter Values

Chapter 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.

Chapter 2. Configuration

This section is normative.

2.1. Transaction Meter Values

Updated in OCPP 2.1

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`
- `SampledDataUpstreamMeasurands`
- `SampledDataUpstreamInterval`

`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).

`SampledDataUpstreamMeasurands` is a comma separated list that prescribes the set of measurands to be included in the *meterValues* field of a TransactionEventRequest (eventType = Started/Updated), every `SampledDataUpstreamInterval` seconds.

WARNING Upstream measurands may be considered to be privacy-sensitive data. As such, this may be subject to regulation and may have legal liability implications. An explicit agreement may be required to allow this.

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

Updated in OCPP 2.1

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`
- `AlignedDataInterval`
- `AlignedDataTxEndedMeasurands`
- `AlignedDataTxEndedInterval`
- `AlignedDataSendDuringIdle`
- `AlignedDataUpstreamMeasurands`
- `AlignedDataUpstreamInterval`

`AlignedDataInterval` 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 `AlignedDataInterval` 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.

- `AlignedDataUpstreamMeasurands`
- `AlignedDataUpstreamInterval`

`AlignedDataSendDuringIdle` can be used to only send clock aligned meter values when there are no ongoing transactions.

NOTE Clock-aligned meter values for an EVSE that is involved in a transaction MAY be transmitted in TransactionEventRequests with *context* = `Sample.Clock` instead of in MeterValuesRequests.

2.3. Multiple Locations/Phases

Updated in OCPP 2.1

When a Charging Station has measurands configured in `SampledDataTxStarted/Updated/EndedMeasurands` and/or `AlignedDataMeasurands/AlignedDataTxEndedMeasurands`, that can be measured on multiple locations or phases, then all possible locations and/or phases SHALL be reported. This does not include location `Upstream`. Measurands to be reported for `Upstream` are configured in SampledData/AlignedDataUpstreamMeasurands.

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`. `AlignedDataInterval` is set to 900 (seconds). Then the Charging Station should send: (every 15 minutes)

- a MeterValuesRequest with: evseId = 0; with 3 SampledValue elements, one per phase with location = Inlet.
- a MeterValuesRequest with: evseId = 1; with 3 SampledValue elements, one per phase with location = Outlet.
- a MeterValuesRequest with: evseId = 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 2.1 supports signed meter values. When a Charging Station supports 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.

2.5. Configuration Examples

Below are a few examples of configurations for transaction-related measurands:

Only sampled energy register values for start/stop at end of transaction

- SampledDataCtrlr.TxStartedMeasurands and TxUpdatedMeasurands are left empty.
- SampledDataCtrlr.TxEndedMeasurands = "Energy.Active.Import.Register"
- SampledDataCtrlr.TxEndedInterval = 0

Values of energy register at start, during and end of transaction

- SampledDataCtrlr.TxStartedMeasurands = "Energy.Active.Import.Register"
- SampledDataCtrlr.TxUpdatedMeasurands = "Energy.Active.Import.Register"
- SampledDataCtrlr.TxUpdatedInterval = 300 (every 5 minutes)
- SampledDataCtrlr.TxEndedMeasurands = "Energy.Active.Import.Register"
- SampledDataCtrlr.TxEndedInterval = 0

Only clock-aligned register values during transaction and start/stop at end

- SampledDataCtrlr.TxStartedMeasurands and TxUpdatedMeasurands are left empty.
- SampledDataCtrlr.TxEndedMeasurands = "Energy.Active.Import.Register"
- SampledDataCtrlr.TxEndedInterval = 0
- AlignedDataCtrlr.Measurands = "Energy.Active.Import.Register"
- AlignedDataCtrlr.Interval = 300 (every 5 minutes)

Chapter 3. Use cases & Requirements

3.1. MeterValues

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
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	**1.** The Charging Station sends a MeterValuesRequest message, for offloading Meter Values to the CSMS. **2.** Upon receipt of a MeterValuesRequest message, the CSMS responds with a MeterValuesResponse message.
5	**Prerequisite(s)**	The Charging Station is configured to send Meter values every XX seconds. No transaction is running.
6	**Postcondition(s)**	**Successful postcondition:** n/a **Failure postcondition:** n/a
7	**Error handling**	n/a
8	**Remark(s)**	The phase field is not applicable to all Measurands. 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. The nature of each sampledValue is determined by the optional Measurand, context, location, unit and phase fields. The optional SignedMeterValue field can contain digitally signed binary meter value data.

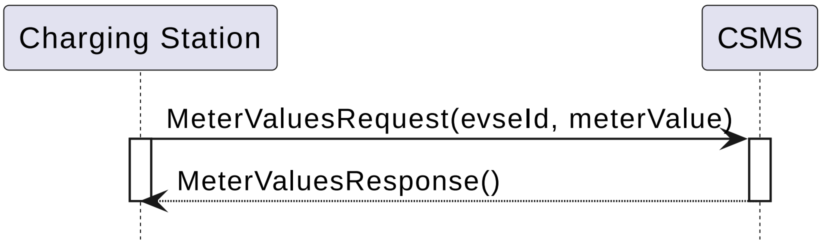


Figure 113. Sequence Diagram: Sending Meter Values

J01 - Sending Meter Values not related to a transaction - Requirements

Table 92. 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 evseId 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 AlignedDataCtrlr.Interval > 0 AND EVSE for which measurands are sent, is not involved in a transaction	The Charging Station SHALL send a MeterValuesRequest message to the CSMS for the measurands in AlignedDataCtrlr.Measurands at every AlignedDataCtrlr.Interval for all *evseIds*, locations and phases for which a configured measurand is supported.	It is possible that certain measurands are not available for every location. For example, *evseId* = 0 (grid meter) will not have a "Current.Offered" or "SoC" measurand. See also J01.FR.22
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.

ID	Precondition	Requirement definition	Note
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.	
J01.FR.21	`AlignedDataSignReadings` is *true*	The Charging Station SHALL retrieve signed meter values from components that support data signing and put them in the *signedMeterValue* field.	
J01.FR.22	When AlignedDataCtrlr.Interval > 0 AND EVSE for which measurands are sent, is involved in a transaction	The Charging Station SHALL send either: - a MeterValuesRequest message or - a TransactionEventRequest with *triggerReason* = `Sample.Clock` to the CSMS for the measurands in AlignedDataCtrlr.Measurands at every AlignedDataCtrlr.Interval.	See also J01.FR.14

J02 - Sending transaction related Meter Values

No.	Type	Description
1	**Name**	Sending transaction related Meter Values
2	**ID**	J02
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	**1.** The Charging Station sends a TransactionEventRequest (eventType = Updated) 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)**	**Successful postcondition:** n/a **Failure postcondition:** n/a
7	**Error handling**	When *Offline*, 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)**	The phase field is not applicable to all Measurands. 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. The nature of each sampledValue is determined by the optional Measurand, context, location, unit and phase fields. The optional SignedMeterValue field can contain digitally signed binary meter value data.

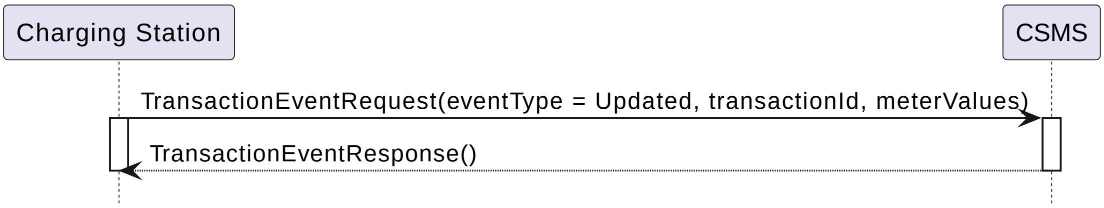

Charging Station
CSMS
TransactionEventRequest(eventType = Updated, transactionId, meterValues)
TransactionEventResponse()

Figure 114. Sequence Diagram: Sending transaction related Meter Values

J02 - Sending transaction related Meter Values - Requirements

Table 93. 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	If a TransactionEventRequest message with eventType = Started or eventType = Update contains multiple *meterValue* elements, rather than one *meterValue* with one or more *sampledValue* elements	All *meterValue* elements SHALL have a timestamp that is within the current sampling interval, i.e.: (transaction event timestamp - `SampledDataTxUpdatedInterval`) < *meterValue.timestamp* <= transaction event timestamp	Only for *eventType* = Ended can a TransactionEventRequest have meter values for multiple intervals.
J02.FR.11	When `SampledDataTxUpdatedInterval` > 0	The Charging Station SHALL send a TransactionEventRequest(eventType = Updated with *triggerReason* = `MeterValuePeriodic` with the measurands configured in SampledDataCtrlr.TxUpdatedMeasurands in the *meterValue* field at every SampledDataCtrlr.TxUpdatedInterval.	See E01 for sending of SampledDataCtrlr.TxStartedMeasurands and E06 for SampledDataCtrlr.TxEndedMeasurands.
J02.FR.12 **(2.1)**	J02.FR.11 AND Charging Station is offline AND Charging Station is running low on memory	The Charging Station MAY drop TransactionEventRequest(eventType = Updated) messages with *triggerReason* = `MeterValueClock` or `MeterValuePeriodic`.	See E12.FR.04 for dropping of other TransactionEvent messages.
J02.FR.13	J02.FR.12	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.	
J02.FR.14	J02.FR.11 AND Amount of meter data is too much for 1 TransactionEventRequest (eventType = Updated)	The Charging Station MAY use multiple TransactionEventRequest(eventType = Updated) messages with the same *timestamp* 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.	
J02.FR.22		Meter values reported in a TransactionEventRequest message SHALL all be related to EVSE on which the transaction is taking place.	
J02.FR.23 **(2.1)**	When a *signedMeterValue* is provided	The field *publicKey*, when present, IS RECOMMENDED to have the format "<marker>:<encoding>:<content-type>:<printed-public-key>" to allow proper decoding, where • <marker> = "oca" • <encoding> = "base16", "base32" or "base64" • <content-type> = "asn1" • <printed-public-key> = <the public key as printed on the certified meter>.	For example: oca:base16:asn1: 3056301006072a8648ce 3d020106052b8104000a 03420004460a02ba2766 d9c44f023ecc0e4e5864 4a87add1aadd6317e5fe 4dccdb29b163a01d8a62 97c84bc530f86431e92f 8d46ab37830247c05cbd 92fac252929e7f61 *(line breaks added for readability)*
J02.FR.24 **(2.1)**	J02.FR.23 AND *signedMeterValue* does not specify the method used to create the digital signature	*signingMethod* SHOULD contain the method used to create the digital signature.	A list of standardized values is provided in the Appendix as SigningMethodEnumStringType.

3.2. ISO 15118 MeterValue signing

J03 - Charging Loop with metering information exchange

No.	Type	Description
1	**Name**	Charging Loop with metering information exchange

No.	Type	Description
2	**ID**	J03
	Reference	ISO15118-1 F1
3	**Objective(s)**	See ISO15118-1, use case Objective F1, page 37.
4	**Description**	See ISO15118-1, use case Description F1, page 37.
	Actors	EV, CSMS, Charging Station
	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.
5	**Prerequisites**	- 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	**Postcondition(s)**	See ISO15118-1, use case End conditions F1, page 37.
7	**Error handling**	n/a
8	**Remark(s)**	n/a

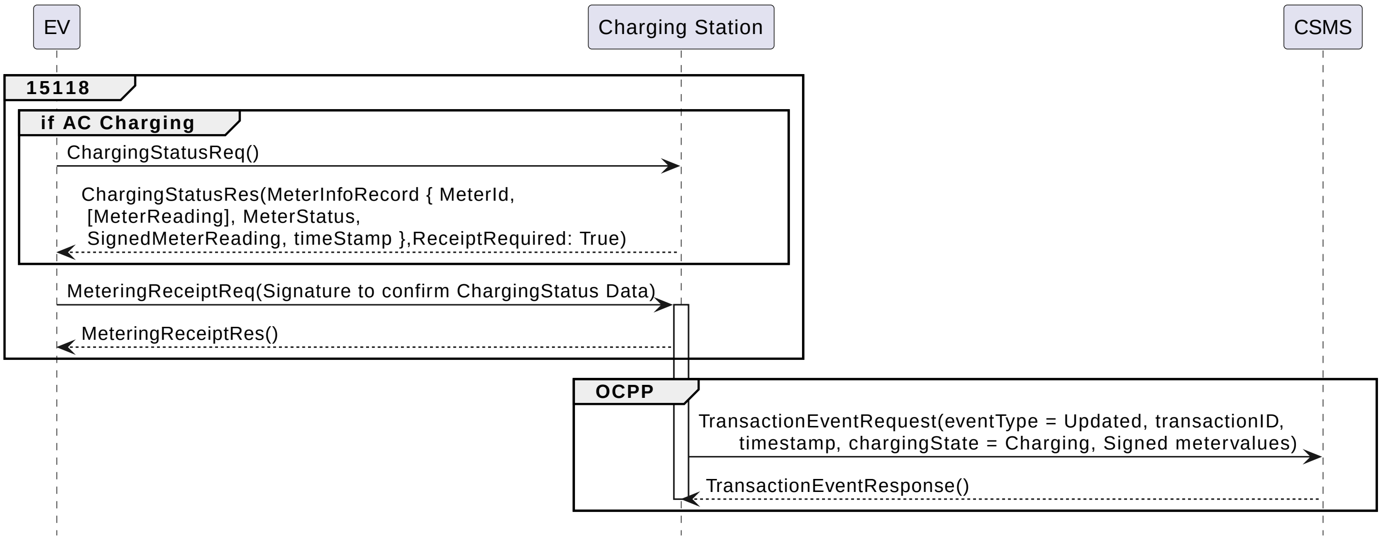

EV
Charging Station
CSMS
15118
if AC Charging
ChargingStatusReq()
ChargingStatusRes(MeterInfoRecord { MeterId, [MeterReading], MeterStatus, SignedMeterReading, timeStamp },ReceiptRequired: True)
MeteringReceiptReq(Signature to confirm ChargingStatus Data)
MeteringReceiptRes()
OCPP
TransactionEventRequest(eventType = Updated, transactionID, timestamp, chargingState = Charging, Signed metervalues)
TransactionEventResponse()

Figure 115. Charging Loop with metering information exchange

J03 - Charging Loop with metering information exchange - Requirements

Table 94. 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 sent to CSMS (as per use case J02) must first have been acknowledged by a MeterReceiptReq from the EV.

K. Smart Charging

Chapter 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

Chapter 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.

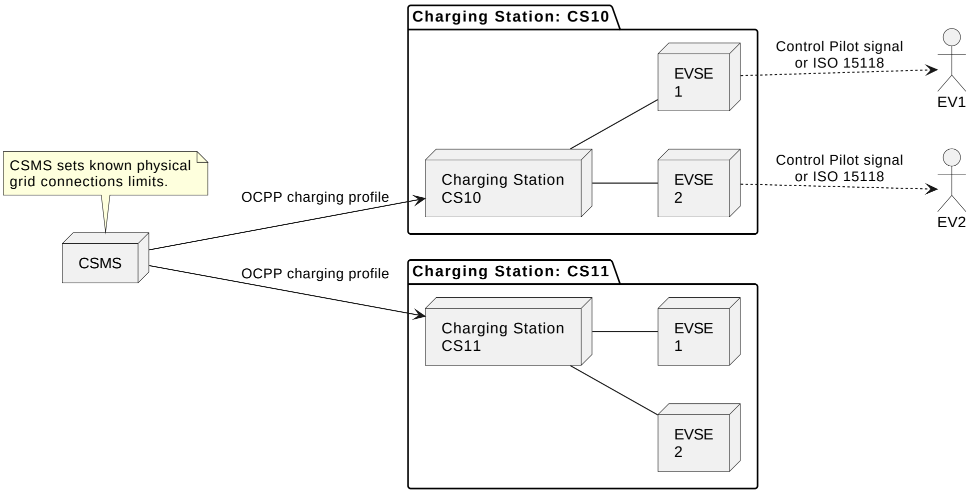


Figure 116. Internal Load Balancing Smart Charging Topology

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.

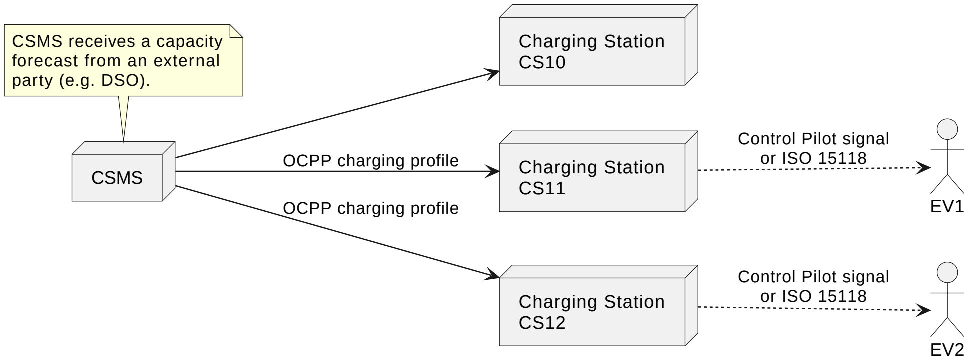

CSMS receives a capacity forecast from an external party (e.g. DSO).
Charging Station CS10
CSMS
OCPP charging profile
Charging Station CS11
Control Pilot signal or ISO 15118
EV1
OCPP charging profile
Charging Station CS12
Control Pilot signal or ISO 15118
EV2

Figure 117. 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 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.

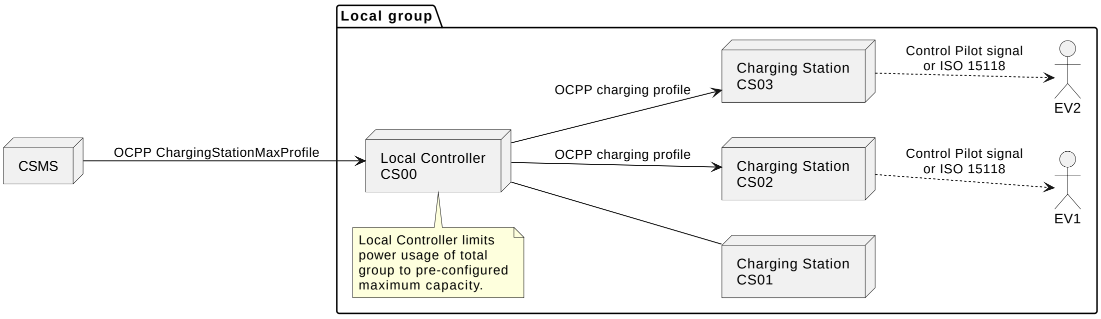

Local group
CSMS
OCPP ChargingStationMaxProfile
Local Controller
CS00
Local Controller limits power usage of total group to pre-configured maximum capacity.
OCPP charging profile
OCPP charging profile
Charging Station
CS03
Charging Station
CS02
Charging Station
CS01
Control Pilot signal or ISO 15118
Control Pilot signal or ISO 15118
EV2
EV1

Figure 118. Local Smart Charging Topology

2.4. External Smart Charging Control Signals

Updated in OCPP 2.1

The OCPP protocol is developed for communication between a CSMS and one or more Charging Stations. As described in the above, this means that a CSMS of a Charging Station Operator (CSO) controls a Charging Station and, based on the charging limits of both the EV and the Charging Station, the CSO controls how fast the EV is charged. In some situations there are other factors that might control charging power: A DSO can send signals to change charging power (e.g. via IEC 61850 [IEC61850-7-420], IEC 60870 [IEC60870-5-104], DNP3 [DNP3] or OpenADR [OPENADR]), or a Home Energy Management System or a smart meter may be in place to limit charging power.

An external actor can connect to a Charging Station with any protocol that is supported by the Charging Station for this purpose, like Modbus, EEBUS, and even OCPP. This control signal can be a single limit value or a schedule. In both cases Charging Station will represent the limit internally as a charging profile of purpose `ChargingStationExternalConstraints`.

A CSMS may need to be informed of changes in charging rate as a result of external signals. OCPP provides a NotifyChargingLimitRequest message to report such changes.

The figures below presents topologies with an Energy Management System, where the external signals (from EMS) are sent directly to the Charging Stations, and a configuration where the signals are sent to a Local Controller.

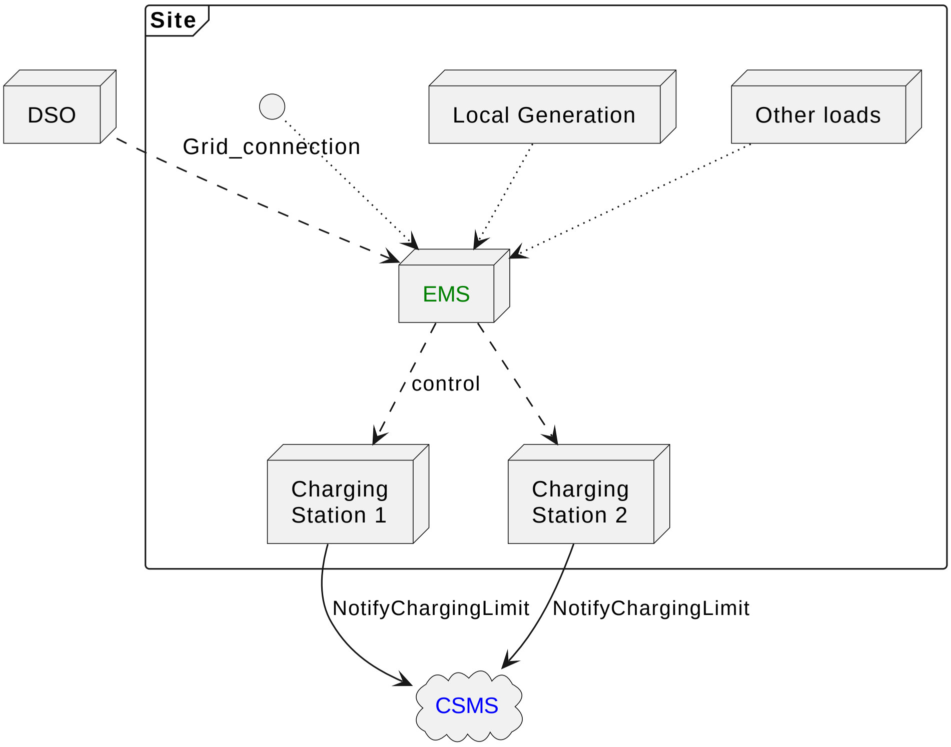


Figure 119. EMS control directly to Charging Stations

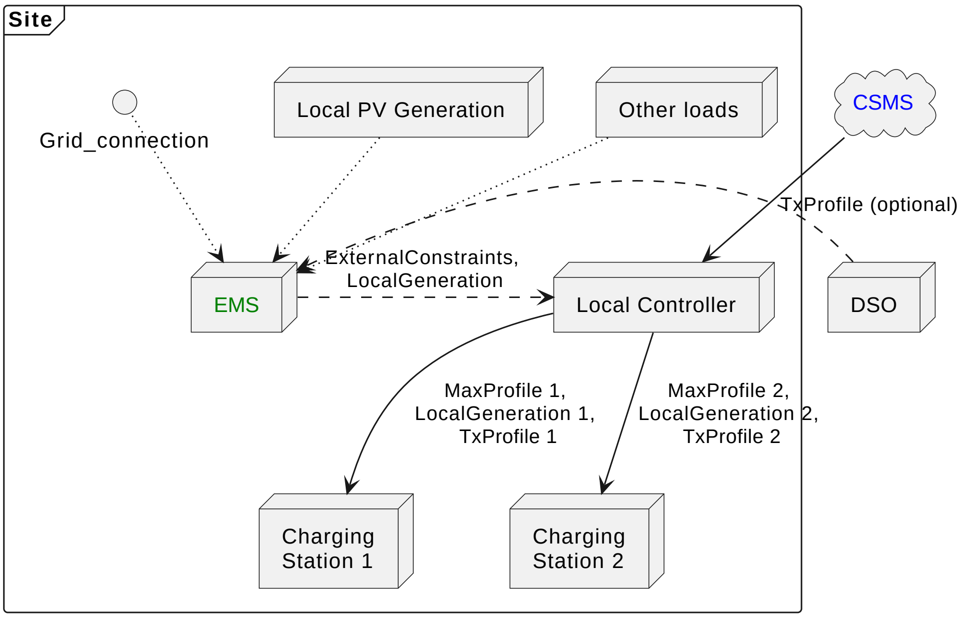


Figure 120. EMS control via Local Controller

See [ref-ocpp21-part1] for a description of common topologies with an EMS.

Chapter 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

Updated in OCPP 2.1

This section describes a number of types of charging profiles that are supported in OCPP. There are six different types of charging profiles, depending on their *purpose*:

ChargingProfile Purpose	**Description**
ChargingStationMaxProfile	A charging profile of purpose `ChargingStationMaxProfile` limits the maximum power or current of the entire Charging Point. A `ChargingStationMaxProfile` is always for evseId = 0. It is kept in persistent storage.
TxProfile	A transaction-specific profile with purpose `TxProfile` 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. It is kept in persistent storage.
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. If the external systems uses OCPP to set the charging profile, then it SHALL use this charging profile purpose when setting the charging profile. A charging schedule in a `ChargingStationExternalConstraints` charging profile is allowed to have schedule periods with *setpoints*, when these periods have *operationMode* = `ExternalSetpoint`. By default, the `ExternalSetpoint` of a `ChargingStationExternalConstraints` profile takes precedence, unless this is overruled by the configuration variable SmartChargingCtrlr.SetpointPriority.
PriorityCharging	A charging profile with purpose `PriorityCharging` is used to overrule the currently active `TxProfile` or `TxDefaultProfile` charging restrictions with a charging profile that provides the maximum possible power under the circumstances, and avoids discharging operations. It has charging schedule periods with *operationMode* = `ChargingOnly` and the charging schedule has no *duration*, since it remains valid until end of the transaction. It is kept in persistent storage.
LocalGeneration	A charging profile that describes locally available power, that **adds** on top of the active charging profiles of a charging station. It is usually originating from an external system, or a local controller, that is aware of the amount of local generation, but can also be provided by a CSMS.

See Limits and setpoints per operation mode for an overview of parameters per *operationMode*.

NOTE A charging profile from an external system is usually received via another protocol than OCPP. The charging station converts the data from the external protocol to an OCPP ChargingProfileType for internal use.

NOTE Charging profile purposes `TxProfile`, `LocalGeneration` and `ChargingStationExternalConstraints`

> are not required to be stored persistently. If any of these purposes is stored persistently by the Charging Station, then it should report this fact in the configuration variable `ChargingProfilePersistence`.

Table 95. operationMode for various ChargingProfilePurposes

ChargingProfilePurpose	**operationMode**
Tx(Default)Profile	ChargingOnly CentralSetpoint ExternalSetpoint ExternalLimits CentralFrequency LocalFrequency LocalLoadBalancing Idle
PriorityCharging	ChargingOnly
ChargingStationMaxProfile	ChargingOnly
ChargingStationExternalConstraints	ChargingOnly ExternalLimits ExternalSetpoint
LocalGeneration	ChargingOnly ExternalLimits

3.3. Charging profile kind

Updated in OCPP 2.1

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 *startSchedule* 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 *startSchedule* is set to a starting point in time. Use this in combination with *recurrencyKind* = `Daily`, 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 the EVSE is ready to deliver energy. i.e. when the EV driver is authorized and the EV is connected (i.e. "PowerPathClosed"). When a ChargingProfile is received for a transaction that is already charging, then the charging schedule periods remain relative to the PowerPathClosed moment of the transaction. No value for *startSchedule* must be supplied.
Dynamic	The schedule consists of only one charging schedule period, in which the limits or setpoints are updated dynamically by CSMS or an external actor. NOTE: This is not related to the ISO 15118-20 dynamic control mode.

3.4. Smart Charging Operation Modes

New in OCPP 2.1

New in OCPP 2.1 is the concept of a smart charging operation mode, that defines a specific mode of operation during a charging schedule period. This operation mode determines for example whether the *setpoint* of a period can be set dynamically by CSMS or an external system.

This mode of operation is defined by the variable *OperationMode* that is part of a ChargingSchedulePeriodType.

The following operation modes exist, of which only the first three operation modes apply to regular smart charging. The others are specific to bidirectional charging. See Introduction.

Unidirectional (charging)

1. **ChargingOnly**
2. **ExternalSetpoint**
3. **ExternalLimits**

Bidirectional (charging/discharging)

1. **CentralSetpoint**
2. **CentralFrequency**
3. **LocalFrequency**
4. **LocalLoadBalancing**
5. **Idle**

ChargingOnly

This operation mode allows charging only. This is also the default operation mode when the field *operationMode* is missing. Behavior of a charging schedule in this mode is identical to OCPP 2.0.1.

The field *limit* is used to specify the maximum allowed charging limit. The fields *dischargeLimit* and *setpoint* are not used.

ExternalSetpoint

This control mode tells the charging station that the *setpoint* parameter is to be determined by some external actor, such as an EMS. A *setpoint* is a charging rate that the EV must try to follow as closely as possible.

If the information for the charging profile comes from the external actor (possibly via a different protocol) then the charging station converts it to a ChargingProfileType with *chargingProfilePurpose* set to `ChargingStationExternalConstraints` and *setpoint* set to the received value.

It is also possible that CSMS submits a Tx(Default)Profile with an *operationMode* = `ExternalSetpoint`. In that case the *setpoint* parameter will be empty. Its value should then be received by the charging station from the external system through some other means of communication and not via OCPP. How this is done will differ for different applications and is out of scope of OCPP.

If the *chargingProfileKind* = `Dynamic`, then the value of *setpoint* can be changed by the external actor during the charging schedule period without having to submit a new charging profile. Whenever the external actor provides a new setpoint, this value is automatically used as the setpoint in the charging schedule period. This is especially suited for rapidly changing conditions.

Optionally the parameter *limit* can be set by CSMS as an upper limit of the range of the external setpoint.

> **NOTE** Only ISO 15118-20 supports the sending of a *setpoint* to an EV.

ExternalLimits

This control mode is similar to ExternalSetpoint with the difference that it is not the *setpoint* that is controlled, but it is the *limit* parameter that is determined by the external actor, such as an EMS.

3.5. 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!

3.6. Combining Charging Profile Purposes

Updated in OCPP 2.1

IMPORTANT Highlighted text in the following paragraphs of this section has been changed.

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* and *setpoint* (taking the different chargingRateUnits into account) among the leading profiles of the different purposes for each time interval. If a charging profile of *chargingProfilePurpose* = `LocalGeneration` is active for the EVSE, then this capacity is **added** on top of the calculated composite schedule.

When a PriorityCharging profile has been activated (either locally or via a UsePriorityChargingRequest), then this will overrule the TxDefaultProfile or TxProfile.

In case the Charging Station is equipped with more than one EVSE, the limit value of ChargingStationMaxProfile for EVSE 0 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.

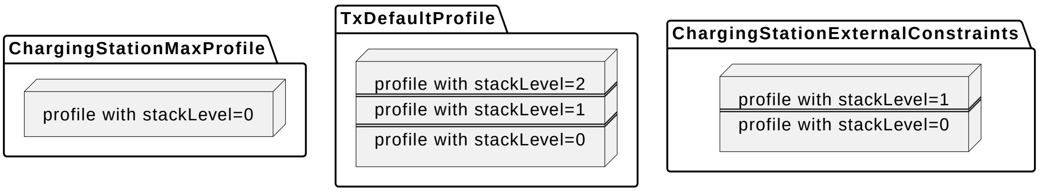

ChargingStationMaxProfile
profile with stackLevel=0
TxDefaultProfile
profile with stackLevel=2
profile with stackLevel=1
profile with stackLevel=0
ChargingStationExternalConstraints
profile with stackLevel=1
profile with stackLevel=0

Figure 121. 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.

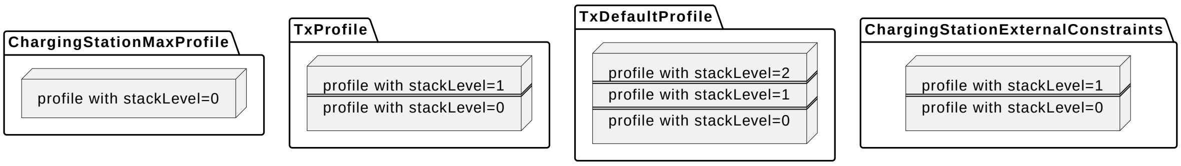

ChargingStationMaxProfile
profile with stackLevel=0
TxProfile
profile with stackLevel=1
profile with stackLevel=0
TxDefaultProfile
profile with stackLevel=2
profile with stackLevel=1
profile with stackLevel=0
ChargingStationExternalConstraints
profile with stackLevel=1
profile with stackLevel=0

Figure 122. 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.

3.7. 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).

3.7.1. 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 or 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`.

3.8. Avoiding Phase Conflicts

In the situation where a `ChargingStationMaxProfile` or a `ChargingStationExternalConstraints` define a value for *numberPhases* or *phaseToUse*, then a possible conflict might arise if such values are also specified in a `TxDefaultProfile` or `TxProfile`. The following rules apply in that case:

numberPhases

The lowest value for a schedule period of all applicable profiles is used for the composite schedule period. If `ChargingStationMaxProfile` has *numberPhases* = 2 or 3 and `TxProfile` has *numberPhases* = 1, then the value 1 is used. The same applies to the reverse situation.

phaseToUse

When there is a conflicting value of *phaseToUse* between the schedule periods of applicable profiles, then there is no way to create a composite schedule period. For example, a CSMS should not submit a charging profile of purpose `ChargingStationMaxProfile` for *phaseToUse* = 1 and then a `TxProfile` for *phaseToUse* = 3, because the charging station will not know which value has preference. Therefore, a SetChargingProfileRequest that causes such a conflict will have to be rejected.
When a relative `TxProfile` is being used and different phases occur in various schedule periods, then it may become difficult to detect if and where such a phase conflict occurs. A charging station should only accept a SetChargingProfileRequest when it can be certain, that there is a no risk of a phase conflict. This means, that when the charging station is not able to verify that no phase conflict occurs in any schedule period (which can happen when the `TxProfile` is received for a transaction, but charging has not yet started, so that start time of the first schedule period is not known), that it cannot accept a charging profile if any of the schedule periods contains a value for *phaseToUse* that differs from the value used in the `ChargingStationMaxProfile` or `ChargingStationExternalConstraints`.

NOTE A value of *phaseToUse* may only be used when *numberOfPhases* = 1.

3.9. Using randomized delays in schedule periods

(New in OCPP 2.1)

A charging profile of type TxDefaultProfile is especially useful to define hours for peak and off-peak charging. Suppose that charging during peak hours is not allowed or has to be at a lower charging rate, then TxDefaultProfile can define a standard schedule for every charging transaction using a list of allowed charging rates over time. The rate is set to zero when no charging is allowed in a period.
The peak/off-peak charging schedule is configured as a so-called "recurring profile": it has a schedule for a period of 24 hours and then repeats every day.

When an EV driver starts a transaction during peak hours, the charging will be suspended until the period of off-peak hours starts, after which charging is allowed normally. If charging still continues when peak hours begin, the energy transfer will be suspended again until end of peak hours. If such a standard TxDefaultProfile is used for a large population of charging stations, for example for all residential chargers in a city or region, then suspending or resuming charging for all those EVs at exactly the same time will have a negative impact on the electric grid.

A charging schedule with a value for *randomizedDelay*, will automatically delay the start of each period with a random time between 0 and *randomizedDelay*. The first charging schedule period will, however, not be delayed to avoid introducing a gap between the start of the charging schedule (at *startSchedule* or start of energy transfer in case of a relative charging profile) and the first period becoming active. The *duration*, when present, will not be affected either.

The above means that *randomizedDelay* will not affect starting and stopping of charging schedules and will not affect recurrence of charging schedules. In case a random delay is required at start of a schedule, then this can be achieved by inserting a short period of *limit* = 0 as the first period with *startPeriod* = 0.

Example 1. Example TxDefaultProfile: peak hours from 8.00h to 11.00h and 16.00h to 22.00h.

```
"chargingProfile": {
  "id": 10, "stackLevel": 1, "chargingProfilePurpose": "TxDefaultProfile",
  "chargingProfileKind": "Recurring", "recurrencyKind": "Daily",
  "chargingSchedule": [ {
    "id": 1, "startSchedule": "2022-06-30T00:00:00",
    "duration": 86400, // 24 hrs
    "chargingRateUnit": "W",
    "randomizedDelay": 600,
    "chargingSchedulePeriod": [
      { "startPeriod": 0, "limit": 999999 },      // 0:00h
      { "startPeriod": 28800, "limit": 0 },       // 8:00h
      { "startPeriod": 39600, "limit": 999999 },  // 11:00h
      { "startPeriod": 57600, "limit": 0 },       // 16:00h
      { "startPeriod": 79200, "limit": 999999 } ] // 22:00h
  } ]
}
```

This defines peak hours as: [28800, 39600] and [57600, 79200] seconds since midnight with a randomized delay, which is calculated for each period at the time the transaction starts as a value between 0 and 600 seconds. This can become something like this:

- for transaction X: [28800+**123**, 39600+**345**] and [57600+**234**, 79200+**456**], and
- for transaction Y: [28800+**555**, 39600+**111**] and [57600+**333**, 79200+**222**]

in which the bold numbers represent the random delays.

Chapter 4. Smart Charging Signals to a Charging Station from Multiple Actors

Updated in OCPP 2.1

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 96. Smart Charging rules for multiple actor situation

ID	Precondition	Requirement definition	Note
SC.01 **(2.1)**		At any point in time, the charging *limit* and *setpoint*, which are the result of merging the schedules from external sources and the CSMS charging profiles with the highest stackLevel from each of the purposes `ChargingStationMaxProfile`, `ChargingStationExternalConstraints` and `TxDefaultProfile` (or `TxProfile` or `PriorityCharging`), SHALL be less than or equal to the lowest value of available power or current in any of the merged schedules. If a charging profile with purpose `LocalGeneration` is active, this capacity is added on top of the calculated charging limit based on the above-mentioned merged schedules.	For safety purposes. For PriorityCharging, see use cases K21, K22. For LocalGeneration, see use case K23.
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 or setpoint in the ChargingProfile for mechanisms 3 and 4 as described in SC.02 MAY be skipped if the change in the limit or setpoint 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 EMS / smart meter input at the Charging Station)

ID	Precondition	Requirement definition	Note
SC.04 **(2.1)**		The GetCompositeScheduleResponse message SHALL always report the expected charging schedule, i.e. the lowest *limit* and *setpoint* when charging or the highest *dischargingLimit* and *setpoint* when discharging. This means that when a transaction has a charging limit X, and EV indicates (via NotifyEVChargingScheduleRequest) that it will use less energy than offered, amount Y, the Charging Station SHALL report limit Y.	Note, that *setpoint* is negative when discharging, and *dischargingLimit* is always negative. (A *setpoint* = -1000 is higher than a *setpoint* = -2000). Therefore, when values are negative, the "highest" value represents the least amount of energy being transferred.

Chapter 5. Use cases & Requirements

5.1. General Smart Charging

K01 - SetChargingProfile

No.	Type	Description
1	**Name**	SetChargingProfile
2	**ID**	K01
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	**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)**	**Successful postcondition:** The Charging Station *Successfully* influences the charging power or current of a specific EV, following the SetChargingProfileRequest sent by the CSMS. **Failure postcondition:** The Charging Station was *not* able to influence the charging power or current of a specific EV, following the SetChargingProfileRequest sent by the CSMS.
7	**Error handling**	n/a
8	**Remark(s)**	n/a

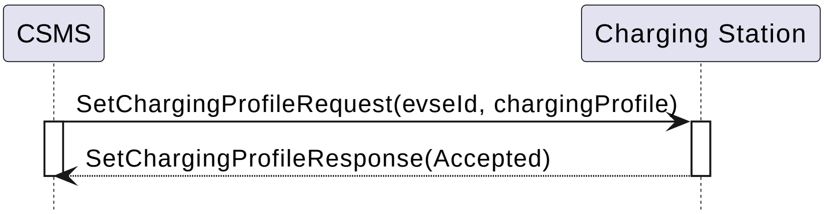

CSMS
Charging Station
SetChargingProfileRequest(evseId, chargingProfile)
SetChargingProfileResponse(Accepted)

Figure 123. Sequence Diagram: SetChargingProfile

K01 - SetChargingProfile - Requirements

IMPORTANT Requirements for charging profile fields *dischargingLimit*, *setpoint* and *setpointReactive* are part of section Introduction.

Table 97. 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 *transactionId* 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 *transactionId* is known	The Charging Station SHALL apply the sent TxProfile to the transaction with the specified *transactionId*.	

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 - evseId* 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`.	It is recommended to include *reasonCode* = "TxNotFound".
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 evseId = 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 evseId > 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 evseId >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`.	It is recommended to include *reasonCode* = "UnsupportedRateUnit".
K01.FR.27 **(2.1)**		ChargingProfiles set via SetChargingProfileRequest with *chargingProfilePurpose* = `ChargingStationMaxProfile`, `TxDefaultProfile` or `PriorityCharging` SHALL be persistent across reboots/power cycles,	Charging profiles `TxProfile`, `LocalGeneration` or `ChargingStation-ExternalConstraints` potentially change often and might eventually wear out persistent memory. They are therefore not required to be persistent.
K01.FR.28	When a SetChargingProfileRequest is received for an evseId that does not exist.	Charging Station SHALL respond with SetChargingProfileResponse with status `Rejected`	It is recommended to include *reasonCode* = "UnknownEVSE".
K01.FR.29	When Charging Station does not support smart charging.	Charging Station SHALL respond with RPC Framework CALLERROR: NotSupported or NotImplemented.	
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) (evseId = 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.

ID	Precondition	Requirement definition	Note
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 >= *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*.	
K01.FR.38	When *chargingProfilePurpose* = `ChargingStationMaxProfile`	*chargingProfileKind* SHALL NOT be `Relative`	
K01.FR.39	When *chargingProfilePurpose* is `TxProfile`	The CSMS SHALL NOT send a ChargingProfile with a *stackLevel* - *transactionId* combination that already exists in another ChargingProfile (with different *id*) 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 *chargingProfileKind* of a ChargingProfile is `Absolute` or `Recurring`	A value for *startSchedule* SHALL exist in the ChargingSchedule of the ChargingProfile.	This determines start date-time of the schedule and of the recurrency sequence.
K01.FR.41	When *chargingProfileKind* of a ChargingProfile is `Relative`	The field *startSchedule* SHALL be absent in the ChargingSchedule of the ChargingProfile.	A relative profile starts from when the profile is activated. (See K01.FR.42)
K01.FR.42 **(2.1)**	K01.FR.41	Charging Station SHALL start the first ChargingSchedulePeriod at the moment the Charging Station is ready to deliver energy, i.e. when the EV driver is authorized and the EV is connected.	If PowerPathClosed is a TxStartPoint, then this will concur with the start of a transaction.
K01.FR.43	When a SetChargingProfileRequest with a value for *numberPhases* is received AND the EVSE is of type AC AND the Charging Station cannot ensure that no more than the received numberPhases will be used	The Charging Station SHALL respond with status = *Rejected*	Is is recommended to include *reasonCode* = "InvalidSchedule". 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 **(2.1)**	When a SetChargingProfileRequest with a value for *numberPhases* or *phaseToUse* is received AND the EVSE is of type DC AND `DCInputPhaseControl` is false or does not exist	The Charging Station SHOULD respond with SetChargingProfileResponse with *status* = `Rejected` and *reasonCode* = "NoPhaseForDC"	A Charging Station may accept the charging profile when it is capable of using the phase information to control the DC EVSE input phases from the grid. See K01.FR.54.

ID	Precondition	Requirement definition	Note
K01.FR.45	When a SetChargingProfileRequest with a value for *numberPhases* is received AND the EVSE is of type AC AND the received *numberPhases* is NOT supported by the Charging Station and higher than the *numberPhases* that are supported by the Charging Station	The Charging Station MAY respond with status = *Accepted*, instead of *Rejected* and impose the limits to a lower *numberPhases*	Please refer to requirement K01.FR.50 and K01.FR.51, for correctly calculating the limits per phase.
K01.FR.46	When a SetChargingProfileRequest with *numberPhases = 1* and a value for *phaseToUse* 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 *true* OR the EVSE is already going to use the received *phaseToUse*	The Charging Station SHALL use the phase indicated by the received *phaseToUse* to connect to the EV.	
K01.FR.47	When a SetChargingProfileRequest with *numberPhases = 1* and *phaseToUse* 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 *phaseToUse* 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 *false* AND the EVSE is NOT going to use the received *phaseToUse*	The Charging Station SHALL respond with status = *Rejected*.	It is recommended to include *reasonCode* = "InvalidSchedule".
K01.FR.49	When a SetChargingProfileRequest without a value for *numberPhases* is received AND the EVSE is of type AC	The Charging Station SHALL assume numberPhases = 3 as a default value.	

ID	Precondition	Requirement definition	Note
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 numberPhases 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.	
K01.FR.52	When a SetChargingProfileRequest with a TxDefaultProfile and *evseId* = 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 *evseId* > 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`.	
K01.FR.54 **(2.1)**	When Charging Station receives a ChargingProfileType with a ChargingSchedulePeriodType with a value for *numberPhases* or *phaseToUse* AND the EVSE is of type DC AND `DCInputPhaseControl` is true	The Charging Station SHALL use *numberPhases* and *phaseToUse* to select the input phases from the grid for the DC EVSE.	See also K01.FR.44
K01.FR.55 **(2.1)**	When Charging Station stores charging profiles of purpose `TxProfile`, `LocalGeneration` or `ChargingStationExternalConstraints` in persistent memory	Charging Station SHOULD report `ChargingProfilePersistence` with the *instance* set to the charging profile purpose, to true.	This tells CSMS that these charging profiles remain present after a reboot.

ID	Precondition	Requirement definition	Note
K01.FR.56 **(2.1)**	When Charging Station receives a SetChargingProfileRequest for a ChargingProfileType with a *chargingProfilePurpose* that is to be stored persistently AND the previous SetChargingProfileRequest for this *chargingProfilePurpose* was less than `ChargingProfileUpdateRateLimit` seconds ago	Charging Station MAY respond with SetChargingProfileResponse with *status* = `Rejected` and *reasonCode* = "RateLimitExceeded"	See also K01.FR.55 and K01.FR.27. If `ChargingProfileUpdateRateLimit` does not exist, there is no rate limit.
		PriorityCharging	
K01.FR.70 **(2.1)**	Charging Station receives a SetChargingProfileRequest with *chargingProfilePurpose* = `PriorityCharging` and a *chargingSchedule* that has a value for *duration*	Charging Station responds with SetChargingProfileResponse with *status* = `Rejected` and an optional *reasonCode* = "InvalidSchedule"	A `PriorityCharging` profile cannot have a *duration*. It lasts until end of transaction, unless it is explicitly ended by EV Driver at the Charging Station or via a request to CSMS.
K01.FR.71 **(2.1)**	Charging Station receives a SetChargingProfileRequest with *chargingProfilePurpose* = `PriorityCharging` and an *operationMode* that is not `ChargingOnly`	Charging Station responds with SetChargingProfileResponse with *status* = `Rejected` and an optional *reasonCode* = "InvalidSchedule"	A `PriorityCharging` profile must only support charging.
		Max External Constraints Id	
K01.FR.80 **(2.1)**	When configuration variable `MaxExternalConstraintsId` is set	CSMS SHALL choose a value for the *id* of a ChargingProfileType that is higher than `MaxExternalConstraintsId`.	This ensures there will not be a conflict with charging profile id's generated at Charging Station to represent `ChargingStationExternalConstraints` profiles.
K01.FR.81 **(2.1)**	When configuration variable `MaxExternalConstraintsId` is set AND Charging Station receives a SetChargingProfileRequest with a *chargingProfile.id* that is less or equal to `MaxExternalConstraintsId`	Charging Station SHALL respond with SetChargingProfileResponse with *status* = `Rejected` and add a *statusInfo* with *reasonCode* = "InvalidProfileId"	
K01.FR.82 **(2.1)**	When configuration variable `MaxExternalConstraintsId` is set AND Charging Station creates a charging profile of *chargingProfilePurpose* = `ChargingStationExternalConstraints`	Charging Station SHALL use a value for *chargingProfile.id* that is less or equal to `MaxExternalConstraintsId`	A Charging Station creates a charging profile `ChargingStationExternalConstraints` to represent limits from an external source, e.g an EMS.
		Use Local Time / Randomized Delay	
K01.FR.90 **(2.1)**	When *useLocalTime* in ChargingScheduleType is true	Charging Station SHALL treat the time in *startSchedule* as an unqualified local time, disregarding any time zone offset or "Z" postfix	What local time is, is configured at Charging Station via `TimeOffset` or `TimeZone`. This allows the same `Absolute` or `Recurring` charging profile to be used in both summer and winter time.

ID	Precondition	Requirement definition	Note
K01.FR.91 **(2.1)**	K01.FR.90 AND A ChargingScheduleType is active at the moment that the local time shifts as a result of beginning or ending of daylight saving time	The execution of the ChargingScheduleType SHALL follow local time, i.e. part of the active *chargingSchedulePeriod(s)* will be skipped or repeated depending on the direction of the change.	This may affect more than one *chargingSchedulePeriod*.
K01.FR.92 **(2.1)**	When *randomizedDelay* > 0 in ChargingScheduleType	Charging Station SHALL postpone the start of each *chargingSchedulePeriod*, except the first one that has *startPeriod* = 0, by a randomly chosen number of seconds between 0 and *randomizedDelay*.	In situations where all charging stations receive the same (recurring) charging profile, this avoids a peak in the grid that would occur when all charging stations change the power limit at exactly the same time. The first period is not delayed to avoid a gap between activation of the charging profile and the first schedule period becoming active. Since the start of the next period defines the end of the current period, this randomization also affects the end of the current period.
K01.FR.93 **(2.1)**	K01.FR.92	Charging Station SHALL calculate random delays for *chargingSchedulePeriods* of the ChargingProfileType at the start of a transaction or when a new ChargingProfileType is submitted during a transaction.	This is to fix the actual start of charging schedule periods during a transaction, such that a GetCompositeScheduleRequest can calculate an accurate composite schedule.
K01.FR.94 **(2.1)**		A *randomizedDelay* SHALL NOT affect the *startSchedule* and *duration* of a ChargingScheduleType.	Only *startPeriod* values of ChargingSchedulePeriodTypes are affected, with the exception of the first ChargingSchedulePeriodType that has *startPeriod* = 0.
K01.FR.95 **(2.1)**	When a SetChargingProfileRequest is received from CSMS with *chargingProfilePurpose* that is NOT (`TxProfile` or `TxDefaultProfile`) AND a *chargingSchedule* with a *randomizedDelay* greater than 0	Charging Station SHALL respond with SetChargingProfileResponse with status `Rejected` and optionally a *reasonCode* = "InvalidSchedule".	From CSMS only Tx(Default)Profiles can have randomized delay. A ChargingStationExternalConstraints charging profile from an external system may also have a randomized delay.
Limit Beyond SoC / Offline validity			
K01.FR.100 **(2.1)**	When *limitAtSoC* is present in ChargingScheduleType and the state of charge of the EV is greater than or equal to *limitAtSoC.soc*	Charging Station SHALL cap the *limit* or *setpoint* in the ChargingSchedulePeriodType by the value of *limitAtSoC.limit*.	This allows to reduce charging power to EVs above a certain SoC. When absent or if SoC measurements are unavailable, Charging Station shall apply the charging schedule without additional limits. Unit of the limit is defined by *chargingRateUnit*.

ID	Precondition	Requirement definition	Note
K01.FR.101 **(2.1)**	When field *maxOfflineDuration* is present in ChargingProfileType AND Charging Station is offline for more than *maxOfflineDuration* seconds	Charging Station SHALL consider this ChargingProfileType to be not valid as long as it remains offline	If a valid charging profile with a lower stack level exists for which *maxOfflineDuration* has not yet expired or is not set, then that will be selected to be used instead.
K01.FR.102 **(2.1)**	When Charging Station's connection is restored	Charging Station SHALL consider the ChargingProfileTypes that had been invalid while offline to be (potentially) valid again, **unless** it has *invalidAfterOfflineDuration* = true, and recalculate which ChargingSchedulePeriodType to execute	It is possible that some charging profiles may have expired during the offline period. A missing *invalidAfterOfflineDuration* is the same as *invalidAfterOfflineDuration* = false.
K01.FR.103 **(2.1)**	When field *maxOfflineDuration* is not present in ChargingProfileType	Charging Station SHALL NOT consider this ChargingProfileType to be invalid during an offline situation.	There may be other reasons why the charging profile becomes invalid, though.
		OperationMode	
K01.FR.110 **(2.1)**	When a new ChargingSchedulePeriodType starts in a ChargingProfileType that applies to this transaction, and the (optional) *operationMode* differs from that of the previous ChargingSchedulePeriodType	Charging Station SHALL send a TransactionEventRequest with *triggerReason* = `OperationModeChanged` and a *transactionInfo* element containing the new *operationMode*.	The default value (when absent) of *operationMode* is `ChargingOnly`.
		Checking optional support	
K01.FR.120 **(2.1)**	When Charging Station receives a SetChargingProfileRequest with a *chargingProfilePurpose* `PriorityCharging` or `LocalGeneration` AND This *chargingProfilePurpose* is not in `SupportedAdditionalPurposes`	Charging Station SHALL respond with SetChargingProfileResponse with *status* = `Rejected` and optionally with *reasonCode* = "UnsupportedPurpose".	
K01.FR.121 **(2.1)**	When Charging Station receives a SetChargingProfileRequest with a *chargingProfileKind* `Dynamic` AND `SupportsDynamicProfiles` is false or absent	Charging Station SHALL respond with SetChargingProfileResponse with *status* = `Rejected` and optionally with *reasonCode* = "UnsupportedKind".	
K01.FR.122 **(2.1)**	When Charging Station receives a SetChargingProfileRequest with a *dynUpdateInterval* or *dynUpdateTime* field AND *chargingProfileKind* is not `Dynamic`	Charging Station SHALL respond with SetChargingProfileResponse with *status* = `Rejected` and optionally with *reasonCode* = "InvalidProfile".	
K01.FR.123 **(2.1)**	When Charging Station receives a SetChargingProfileRequest with *useLocalTime* = true AND `SupportsLocalTime` is false or absent	Charging Station SHALL respond with SetChargingProfileResponse with *status* = `Rejected` and optionally with *reasonCode* = "InvalidSchedule".	

ID	Precondition	Requirement definition	Note
K01.FR.124 **(2.1)**	When Charging Station receives a SetChargingProfileRequest with *randomizedDelay* > 0 AND `SupportsRandomizedDelay` is false or absent	Charging Station SHALL respond with SetChargingProfileResponse with *status* = `Rejected` and optionally with *reasonCode* = "InvalidSchedule".	
K01.FR.125 **(2.1)**	When Charging Station receives a SetChargingProfileRequest with field *limitAtSoC* AND `SupportsLimitAtSoC` is false or absent	Charging Station SHALL respond with SetChargingProfileResponse with *status* = `Rejected` and optionally with *reasonCode* = "InvalidSchedule".	
K01.FR.126 **(2.1)**	When Charging Station receives a SetChargingProfileRequest with *evseSleep* = true AND `SupportsEvseSleep` is false or absent	Charging Station SHALL respond with SetChargingProfileResponse with *status* = `Rejected` and optionally with *reasonCode* = "InvalidSchedule".	

K02 - Central Smart Charging

No.	Type	Description
1	**Name**	Central Smart Charging
2	**ID**	K02
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 the EV. The CSMS calculates a ChargingSchedule to stay within limits which MAY be imposed by any external system. See: *Central Smart Charging*
	Actors	Charging Station, CSMS, EV, EV Driver
	Scenario description	**1.** 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. **2.** The EV starts charging and a TransactionEventRequest is sent to the CSMS. **3.** The CSMS responds with a TransactionEventResponse. **4.** In response to a TransactionEventRequest the CSMS MAY choose to set charging limits to the transaction using a SetChargingProfileRequest. **5.** The Charging Station responds with a SetChargingProfileResponse. **6.** While charging is in progress the EVSE will continuously adapt the maximum current or power according to the installed ChargingProfiles.
	Alternative scenario(s)	K03 - Local Smart Charging K04 - Internal Load Balancing
5	**Prerequisite(s)**	The Functional Block *Smart Charging* is installed.
6	**Postcondition(s)**	**Successful postcondition:** The Charging Station *Successfully* influences the charging power or current of a specific EV, following the SetChargingProfileRequest sent by the CSMS. **Failure postcondition:** The Charging Station was *not* able to influence the charging power or current of a specific EV, following the SetChargingProfileRequest sent by the CSMS.
7	**Error handling**	n/a

No.	Type	Description
8	**Remark(s)**	The CSMS determines the constraints on ChargingSchedule per transaction. 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 `offline` 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 *Offline* period. 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. 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. The scenario description and sequence diagram above are not based on a specific `TxStartPoint`. The `TxStopPoint` is: 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 send. For more details see the use case: E01 - Start Transaction options.

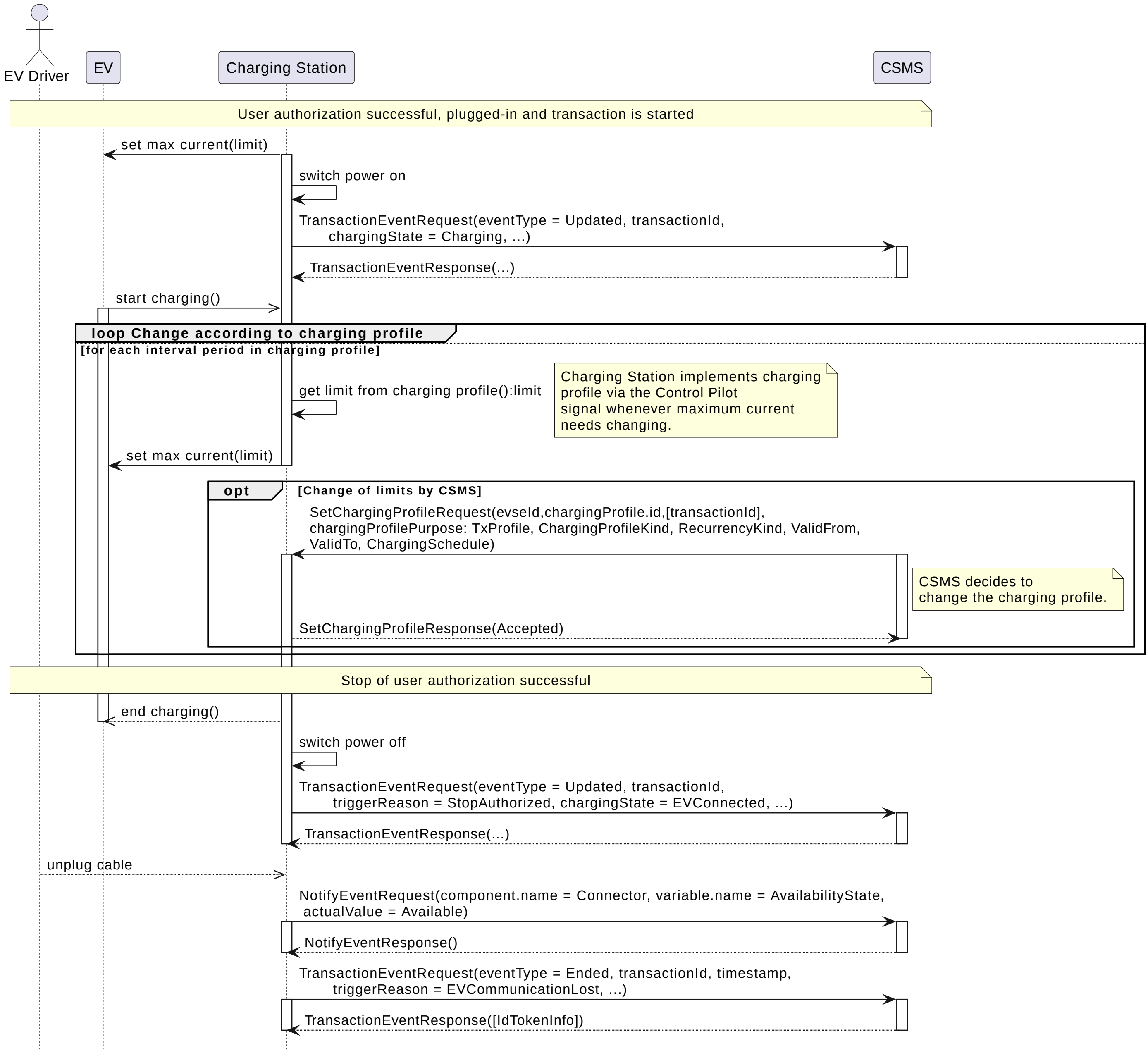

EV Driver
EV
Charging Station
CSMS
User authorization successful, plugged-in and transaction is started
set max current(limit)
switch power on
TransactionEventRequest(eventType = Updated, transactionId, chargingState = Charging, ...)
TransactionEventResponse(...)
start charging()
loop Change according to charging profile
[for each interval period in charging profile]
get limit from charging profile():limit
Charging Station implements charging profile via the Control Pilot signal whenever maximum current needs changing.
set max current(limit)
opt
[Change of limits by CSMS]
SetChargingProfileRequest(evseId,chargingProfile.id,[transactionId], chargingProfilePurpose: TxProfile, ChargingProfileKind, RecurrencyKind, ValidFrom, ValidTo, ChargingSchedule)
CSMS decides to change the charging profile.
SetChargingProfileResponse(Accepted)
Stop of user authorization successful
end charging()
switch power off
TransactionEventRequest(eventType = Updated, transactionId, triggerReason = StopAuthorized, chargingState = EVConnected, ...)
TransactionEventResponse(...)
unplug cable
NotifyEventRequest(component.name = Connector, variable.name = AvailabilityState, actualValue = Available)
NotifyEventResponse()
TransactionEventRequest(eventType = Ended, transactionId, timestamp, triggerReason = EVCommunicationLost, ...)
TransactionEventResponse([IdTokenInfo])

Figure 124. 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.

K02 - Central Smart Charging - Requirements

Table 98. 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.	

ID	Precondition	Requirement definition	Note
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. (Same as K01.FR.21)
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. (Same as K01.FR.22)
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

No.	Type	Description
1	**Name**	Local Smart Charging
2	**ID**	K03
3	**Objective(s)**	To illustrate the process of local load-balancing by a Local Cluster.
4	**Description**	This use case is an example of how local load-balancing can be performed. It does not imply that other approaches would not be correct. The process has been simplified for clarity and should not be regarded as prescriptive. A Local Controller is configured with a value for maximum current for the total cluster by CSMS via a charging profile of type `ChargingStationMaxProfile` to the Local Controller, or an EMS may have set a `ChargingStationExternalConstraints` charging profile. The Local Controller divides the maximum current among the active transactions. Whenever a transaction starts or finishes, the Local Controller will update the charging profiles of the remaining transactions to divide the maximum current equally. For simplicity’s sake, this use case does not differentiate on departure time or state of charge of vehicles, nor does it take the actual energy consumption of vehicles into account.
	Actors	Charging Station (CS01, CS02), Local Controller (LC), CSMS
	Scenario description	Assume no transactions are active in the local cluster and the maximum current for the local cluster has been configured to be 100 A. The charging stations all have a `TxDefaultProfile` that allows a current of only 6 A, so that vehicles cannot immediately start charging at full power before the LC had the chance to set a charging profile. 1. A transaction starts on charging station CS01. It sends a TransactionEventRequest(`Started`) to LC. 2. LC is configured to do local load-balancing (i.e. its SmartChargingCtrlr.Enabled = true), so it registers the transaction id TX1 of the transaction that has been started on CS01, before forwarding the message on the websocket for CS01 towards CSMS. 3. LC sends a SetChargingProfileRequest to CS01 with *chargingProfilePurpose* = `TxProfile`, *chargingProfileKind* = `Relative`, *transactionId* = TX1 and a *chargingSchedule* with a *chargingRateUnit* = `A` that contains one *chargingSchedulePeriod* with a *limit* of 94 A, so that the entire quota is available to this transaction minus the `TxDefaultProfile` amount for new transactions. 4. Another transaction starts on charging station CS02. It sends a TransactionEventRequest(`Started`) to LC. 5. LC registers the new transaction id TX2 and forwards the message on the websocket for CS02 to CSMS. 6. LC divides the available quota by allowing each transaction a maximum of 47 A. 7. LC sends a SetChargingProfile message to CS01 that updates the existing `TxProfile` and sets the *limit* to 47 A. 8. LC sends new SetChargingProfile to CS02 with *chargingProfilePurpose* = `TxProfile`, *chargingProfileKind* = `Relative`, *transactionId* = TX2 and a *chargingSchedule* with a *chargingRateUnit* = `A` that contains one *chargingSchedulePeriod* with a *limit* of 47 A. 9. The transaction of CS01 finishes. It sends a TransactionEventRequest(`Ended`) to LC. 10. LC registers that transaction TX1 on CS01 has finished and forwards the message on the websocket for CS01 to CSMS. 11. LC now allows the maximum to TX2. It sends a SetChargingProfile message to CS02 that updates the existing `TxProfile` and sets the *limit* to 94 A. (Note, that the `TxProfile` for TX1 on CS01 has automatically ceased to exist upon termination of the transaction.)
5	**Prerequisites**	The LC has been configured with a fixed maximum current level. The SmartChargingCtrlr component of Local Controller has been Enabled, which will trigger the Local Controller to read and interpret TransactionEventRequest messages from connected Charging Stations.
6	**Post conditions**	
7	**Error Handling**	

No.	Type	Description
8	**Remarks**	As described in Part 1, a Local Controller replicates all websockets from Charging Stations in the cluster towards CSMS, and forwards messages from Charging Station to CSMS on the appropriate websocket (and vice versa). This allows the Local Controller to read messages, such as a TransactionEventRequest message, from the Charging Station. 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. The Local Controller MAY or MAY NOT have any EVSEs of its own. 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.

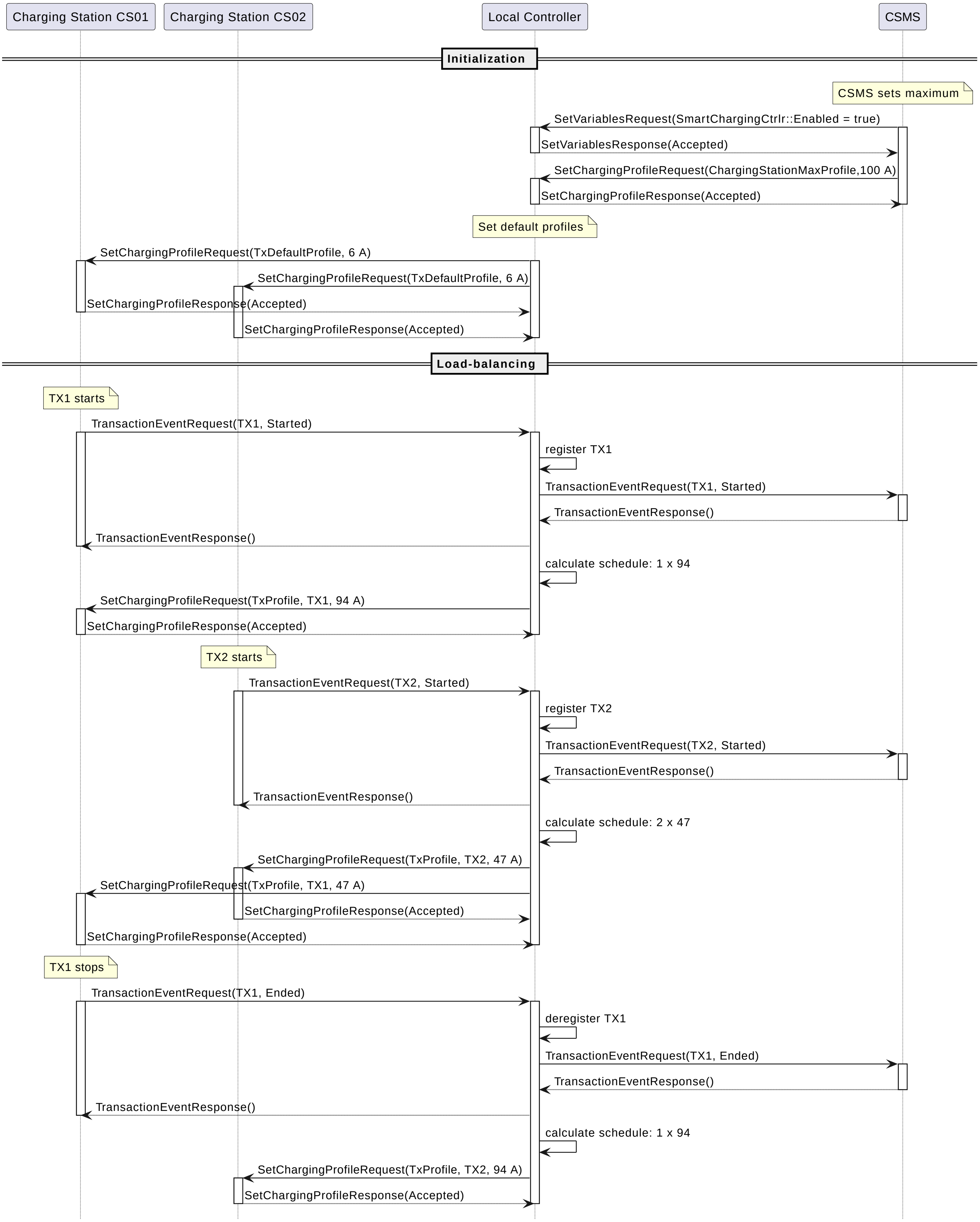

Charging Station CS01
Charging Station CS02
Local Controller
CSMS
Initialization
CSMS sets maximum
SetVariablesRequest(SmartChargingCtrlr::Enabled = true)
SetVariablesResponse(Accepted)
SetChargingProfileRequest(ChargingStationMaxProfile,100 A)
SetChargingProfileResponse(Accepted)
Set default profiles
SetChargingProfileRequest(TxDefaultProfile, 6 A)
SetChargingProfileRequest(TxDefaultProfile, 6 A)
SetChargingProfileResponse(Accepted)
SetChargingProfileResponse(Accepted)
Load-balancing
TX1 starts
TransactionEventRequest(TX1, Started)
register TX1
TransactionEventRequest(TX1, Started)
TransactionEventResponse()
TransactionEventResponse()
calculate schedule: 1 x 94
SetChargingProfileRequest(TxProfile, TX1, 94 A)
SetChargingProfileResponse(Accepted)
TX2 starts
TransactionEventRequest(TX2, Started)
register TX2
TransactionEventRequest(TX2, Started)
TransactionEventResponse()
TransactionEventResponse()
calculate schedule: 2 x 47
SetChargingProfileRequest(TxProfile, TX2, 47 A)
SetChargingProfileRequest(TxProfile, TX1, 47 A)
SetChargingProfileResponse(Accepted)
SetChargingProfileResponse(Accepted)
TX1 stops
TransactionEventRequest(TX1, Ended)
deregister TX1
TransactionEventRequest(TX1, Ended)
TransactionEventResponse()
TransactionEventResponse()
calculate schedule: 1 x 94
SetChargingProfileRequest(TxProfile, TX2, 94 A)
SetChargingProfileResponse(Accepted)

Figure 125. Local Controller performing local load-balancing

K03 - Local Smart Charging - Requirements

Table 99. K03 - Requirements

ID	Precondition	Requirement definition	Note
K03.FR.01		The Local Controller MAY impose charging limits on a Charging Station.	

ID	Precondition	Requirement definition	Note
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. (Same as K01.FR.21)
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. (Same as K01.FR.22)

K04 - Internal Load Balancing

No.	Type	Description
1	**Name**	Internal Load Balancing
2	**ID**	K04
3	**Objective(s)**	To enable internal load balancing within the Charging Station and between EVSEs.
4	**Description**	The Load Balancing use case is about 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. See K01 - Set Charging Profile
	Actors	Charging Station, CSMS
	Scenario description	**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 *Smart Charging* is installed.
6	**Postcondition(s)**	**Successful postcondition:** The Charging Station *Successfully* balances the current/power between the different EVSEs, based on what the CSMS is sending. **Failure postcondition:** ChargingProfile is *not Accepted*. Charging is possible, although the Charging Station will *not* adhere to the ChargingProfile.
7	**Error handling**	n/a
8	**Remark(s)**	n/a

K04 - Internal Load Balancing - Requirements

Table 100. 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.	EVSE 0 refers to entire Charging Station.
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. (Same as K01.FR.21)
K04.FR.05 **(2.1)**		The combined energy flow of all EVSEs (and the Charging Station hardware itself) SHALL NOT be greater than the combined limit set by `ChargingStationMaxProfile` + `LocalGeneration`.	For details about `LocalGeneration` see use case K27 - Smart Charging with EMS and LocalGeneration.

K05 - Remote Start Transaction with Charging Profile

No.	Type	Description
1	**Name**	Remote Start Transaction with Charging Profile
2	**ID**	K05
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 *Offline* 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	**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 *Smart Charging* is installed.
6	**Postcondition(s)**	**Successful postcondition:** The Charging Station *Successfully* charges taking into account the provided ChargingProfile. **Failure postcondition:** The transaction is *not* started. The Charging Station *Unsuccessfully* charges taking into account the provided ChargingProfile.
7	**Error handling**	n/a
8	**Remark(s)**	The scenario description and sequence diagram above are based on the Configuration Variable for start transaction being configured as follows: `TxStartPoint`: EVConnected or PowerPathClosed 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. When a ChargingProfile with purpose TxProfile is provided as part of a RequestStartTransactionRequest, then a *transactionId* cannot be provided in the ChargingProfile, because it is not known at the time.

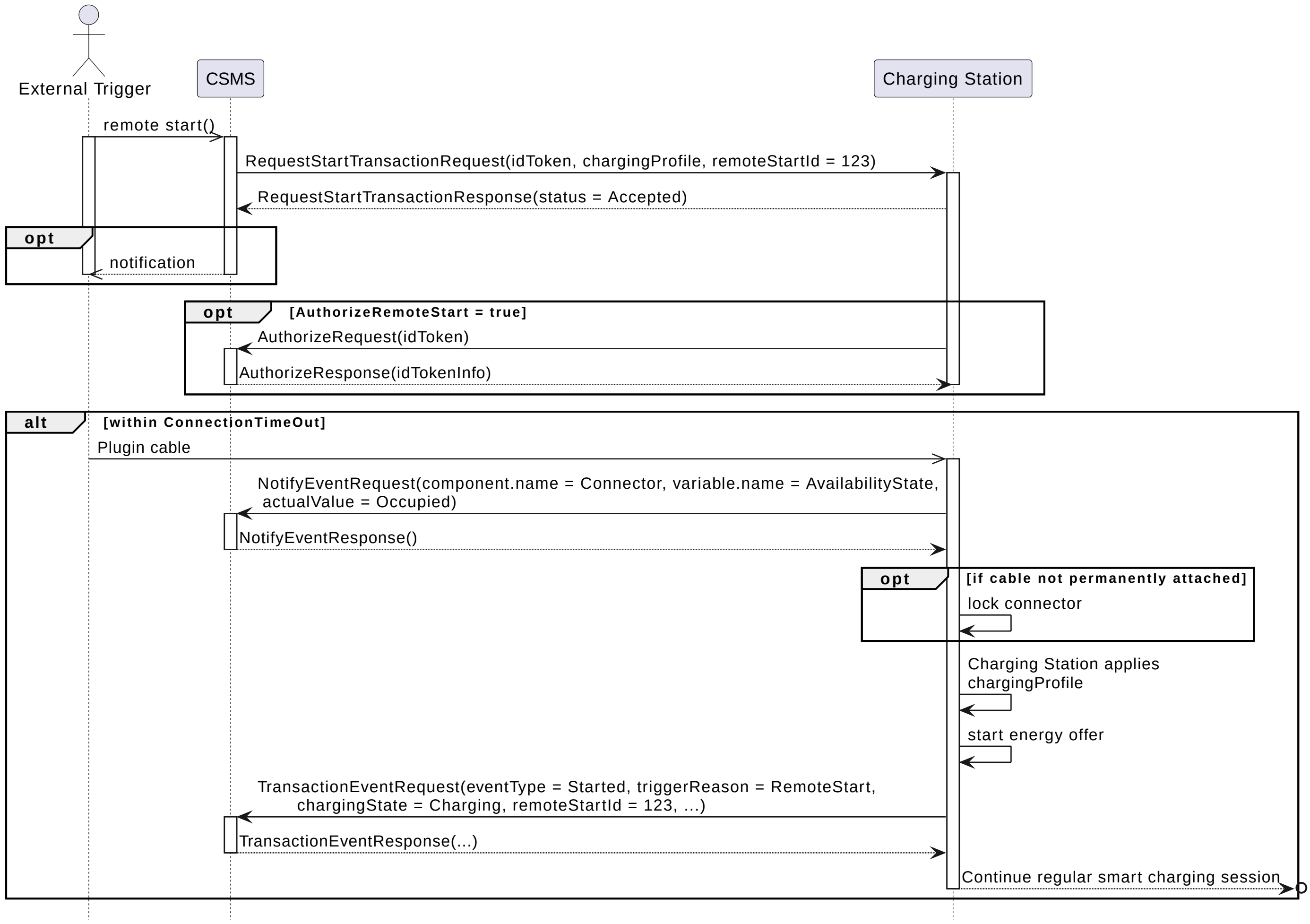

External Trigger
CSMS
Charging Station
remote start()
RequestStartTransactionRequest(idToken, chargingProfile, remoteStartId = 123)
RequestStartTransactionResponse(status = Accepted)
opt
notification
opt
[AuthorizeRemoteStart = true]
AuthorizeRequest(idToken)
AuthorizeResponse(idTokenInfo)
alt
[within ConnectionTimeOut]
Plugin cable
NotifyEventRequest(component.name = Connector, variable.name = AvailabilityState, actualValue = Occupied)
NotifyEventResponse()
opt
[if cable not permanently attached]
lock connector
Charging Station applies chargingProfile
start energy offer
TransactionEventRequest(eventType = Started, triggerReason = RemoteStart, chargingState = Charging, remoteStartId = 123, ...)
TransactionEventResponse(...)
Continue regular smart charging session

Figure 126. Sequence Diagram: Remote Start Transaction with Charging Profile

K05 - Remote Start Transaction with Charging Profile - Requirements

Table 101. 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. Same as F01.FR.12
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. Same as F01.FR.26

K06 - Offline Behavior Smart Charging During Transaction

No.	Type	Description
1	**Name**	Offline Behavior Smart Charging During Transaction

No.	Type	Description
2	**ID**	K06
3	**Objective(s)**	To enable the Charging Station to continue to use the current ChargingProfile for the duration of the transaction while it is *Offline*.
4	**Description**	If a Charging Station goes *Offline* 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	**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 *Offline* 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 *Smart Charging* is installed.
6	**Postcondition(s)**	**Successful postcondition:** The Charging Station continues to use the charging profiles which are available. **Failure postcondition:** n/a
7	**Error handling**	n/a
8	**Remark(s)**	n/a

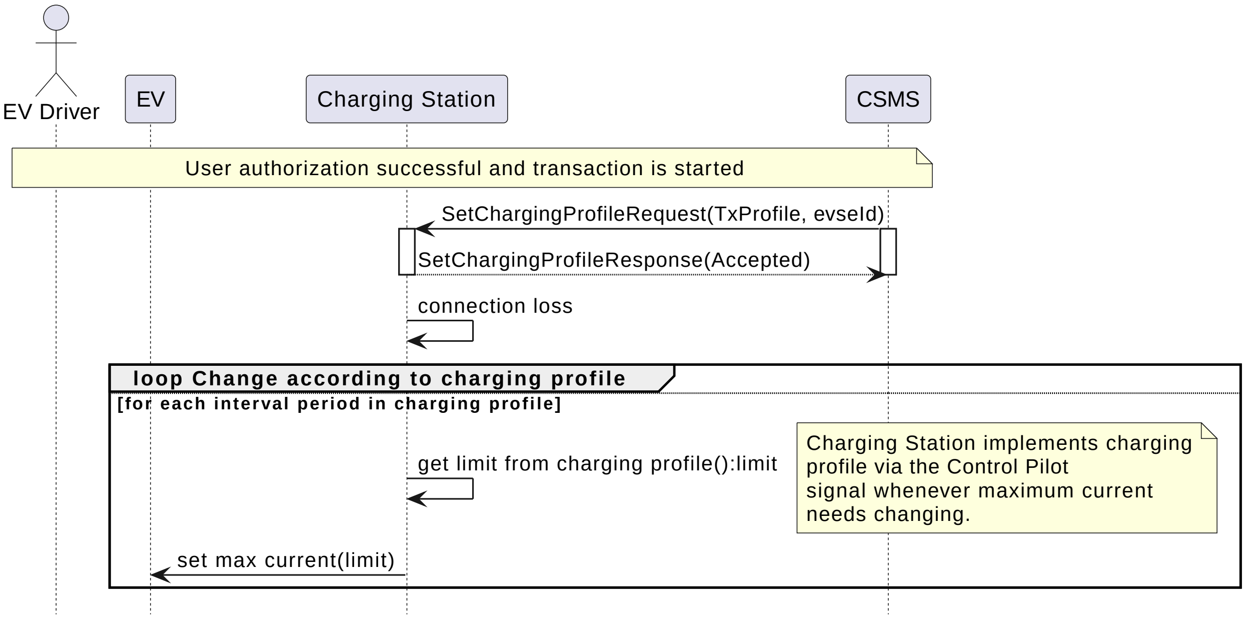

EV Driver
EV
Charging Station
CSMS
User authorization successful and transaction is started
SetChargingProfileRequest(TxProfile, evseId)
SetChargingProfileResponse(Accepted)
connection loss
loop Change according to charging profile
[for each interval period in charging profile]
get limit from charging profile():limit
Charging Station implements charging profile via the Control Pilot signal whenever maximum current needs changing.
set max current(limit)

Figure 127. Sequence Diagram: Offline Behavior Smart Charging

K06 - Offline Behavior Smart Charging During Transaction - Requirements

Table 102. K06 - Requirements

ID	Precondition	Requirement definition
K06.FR.01	If the Charging Station goes *Offline* 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 *Offline*, 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

No.	Type	Description
1	**Name**	Offline Behavior Smart Charging at Start of Transaction
2	**ID**	K07
3	**Objective(s)**	To enable the Charging Station to continue to use a ChargingProfile for a transaction which is started *Offline*.
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 *Offline*, uses this profile.
	Actors	Charging Station, CSMS, EV, EV Driver
	Scenario description	**1.** The CSMS sends a SetChargingProfileRequest to the Charging Station with a TxDefaultProfile. **2.** The Charging Station responds with a SetChargingProfileResponse. **3.** The Charging Station goes *Offline* and operates stand-alone. **4.** The Charging Station allows automatic authorization of any presented IdToken by either: *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 **5.** The transaction is started in the same way as described in E. Transactions. **6.** While charging is in progress the EVSE will continuously adapt the maximum current or power according to the already installed ChargingProfiles.
5	**Prerequisite(s)**	The Charging Station is *Offline*. The Functional Block *Smart Charging* is installed. The IdToken is known in the Local Authorization List, the IdToken is known in the Authorization Cache, or unknown offline authorization is enabled.
6	**Postcondition(s)**	**Successful postcondition:** The Charging Station uses the installed TxDefaultProfile which are available for the *Offline* started transaction. **Failure postcondition:** n/a
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.

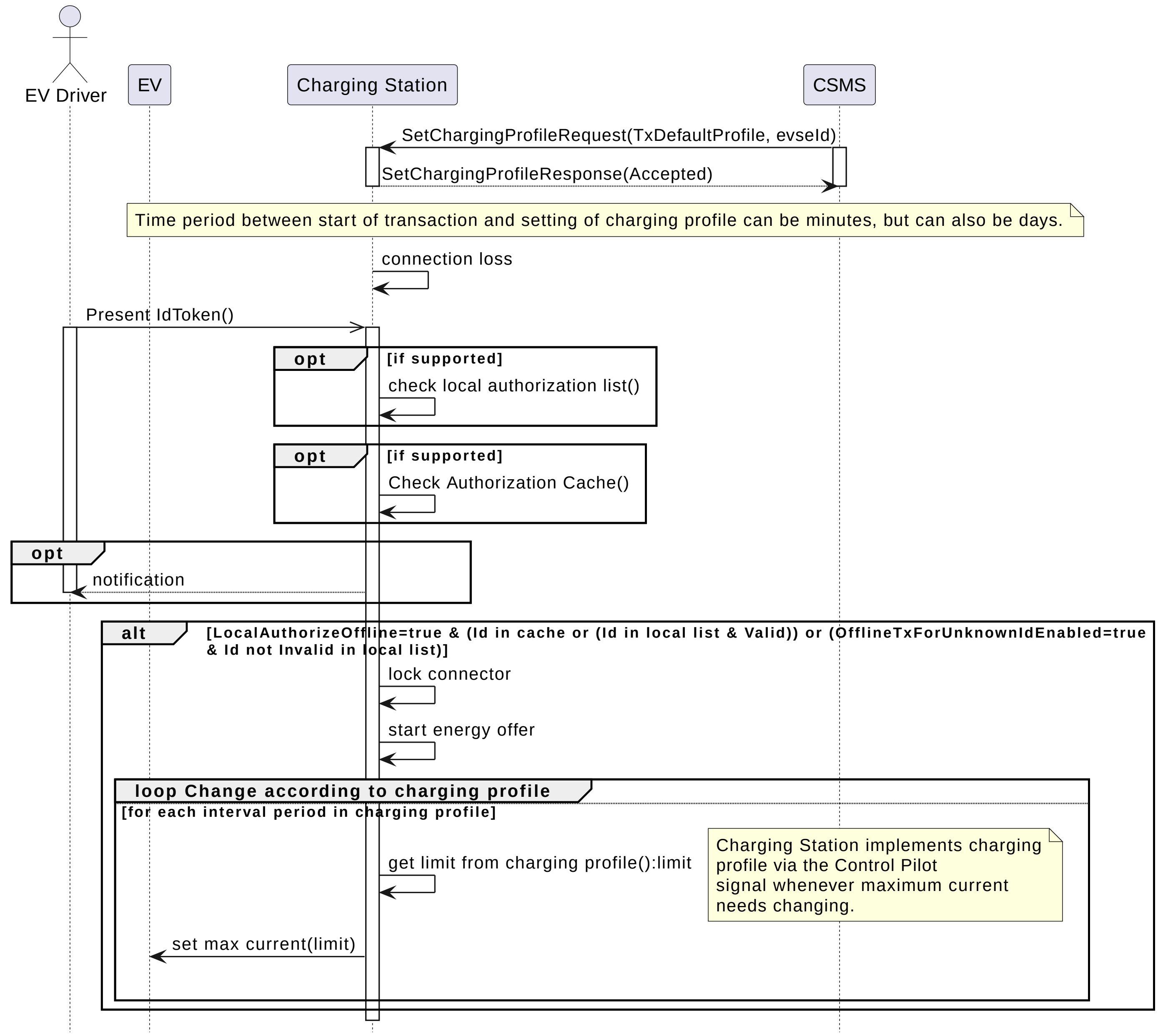

EV Driver
EV
Charging Station
CSMS
SetChargingProfileRequest(TxDefaultProfile, evseId)
SetChargingProfileResponse(Accepted)
Time period between start of transaction and setting of charging profile can be minutes, but can also be days.
connection loss
Present IdToken()
opt
[if supported]
check local authorization list()
opt
[if supported]
Check Authorization Cache()
opt
notification
alt
[LocalAuthorizeOffline=true & (Id in cache or (Id in local list & Valid)) or (OfflineTxForUnknownIdEnabled=true & Id not Invalid in local list)]
lock connector
start energy offer
loop Change according to charging profile
[for each interval period in charging profile]
get limit from charging profile():limit
Charging Station implements charging profile via the Control Pilot signal whenever maximum current needs changing.
set max current(limit)

Figure 128. Sequence Diagram: Offline Behavior Smart Charging

K07 - Offline Behavior Smart Charging at Start of Transaction - Requirements

Table 103. K07 - Requirements

ID	Precondition	Requirement definition	Note
K07.FR.01	If a Charging Station goes *Offline* 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

No.	Type	Description
1	**Name**	Get Composite Schedule
2	**ID**	K08
3	**Objective(s)**	To request the Charging Station to report the composite charging schedule.

No.	Type	Description
4	**Description**	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. The CompositeSchedule is the result of the calculation of all active schedules and possible local limits present in the Charging Station.
	Actors	Charging Station, CSMS
	Scenario description	**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 *Smart Charging* is installed.
6	**Postcondition(s)**	**Successful postcondition:** The CSMS *Successfully* received the composite schedule from the Charging Station. **Failure postcondition:** The CSMS did *not* receive the composite schedule from the Charging Station.
7	**Error handling**	n/a
8	**Remark(s)**	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). The Composite Schedule that will guide the charging level is a combination of the prevailing Charging Profiles of the different chargingProfilePurposes. 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. The reported schedule, in GetCompositeScheduleResponse, is the result of the calculation of all active schedules and possible local limits present in the Charging Station. 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. When requested for evseid=0, the Charging Station will calculate the total expected consumption for the grid connection.

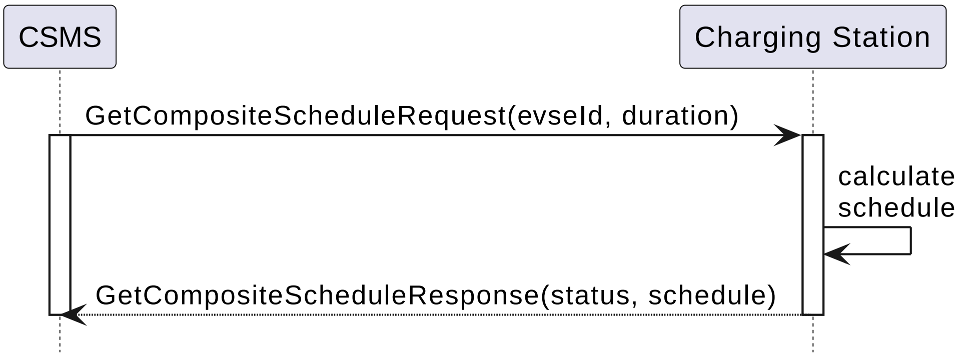

CSMS
Charging Station
GetCompositeScheduleRequest(evseId, duration)
calculate schedule
GetCompositeScheduleResponse(status, schedule)

Figure 129. Sequence Diagram: Get Composite Schedule

K08 - Get Composite Schedule - Requirements

Table 104. K08 - Requirements

ID	Precondition	Requirement definition
K08.FR.01		The CSMS MAY request the Charging Station to report the CompositeSchedule by sending GetCompositeScheduleRequest.

ID	Precondition	Requirement definition
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 evseId 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 **(2.1)**		At any point in time, the charging *limit*, *dischargingLimit* or *setpoint*, *reactiveSetpoint* in the CompositeSchedule which are the result of merging the schedules from all valid charging profiles with the highest stackLevel from each of the purposes `ChargingStationMaxProfile`, `ChargingStationExternalConstraints` and `TxDefaultProfile` (or `TxProfile` or `PriorityCharging`), SHALL be less than or equal to the lowest value of available power or current in any of the merged schedules. If a charging profile with purpose `LocalGeneration` is active, this capacity is added on top of the calculated charging limit based on above-mentioned merged schedules. (See also SC.01)
K08.FR.05	If the Charging Station is not able to report the requested schedule, for instance if the *evseId* is unknown	The Charging Station SHALL respond with the status `Rejected`.
K08.FR.06 **(2.1)**	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) with a *randomizedDelay* = 0.
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`.
K08.FR.08 **(2.1)**	K08.FR.04 AND the result of merging schedules from all charging profiles contains *limit* or *dischargingLimit* values as well as *setpoint* values	Charging Station SHALL cap negative values of *setpoint* to the value of *dischargingLimit* and positive values of *setpoint* to the value of *limit*. (See V2X.03 and V2X.04)

K09 - Get Charging Profiles

No.	Type	Description
1	**Name**	Get Charging Profile
2	**ID**	K09
3	**Objective(s)**	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 install 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
	Scenario description	**1** The CSMS asks the Charging Station for the installed charging profiles by sending a GetChargingProfilesRequest message. **2** The Charging Station responds, indicating if it can report Charging Schedules by sending a GetChargingProfilesResponse message. **3** Charging Station sends a number of ReportChargingProfilesRequest messages to CSMS. **4** The CSMS acknowledges reception of the reports by sending a ReportChargingProfilesResponse to the Charging Station for every ReportChargingProfilesRequest.
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.

No.	Type	Description
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 to much data for one message. To indicate that more reports will follow the flag **tbc** can be used.

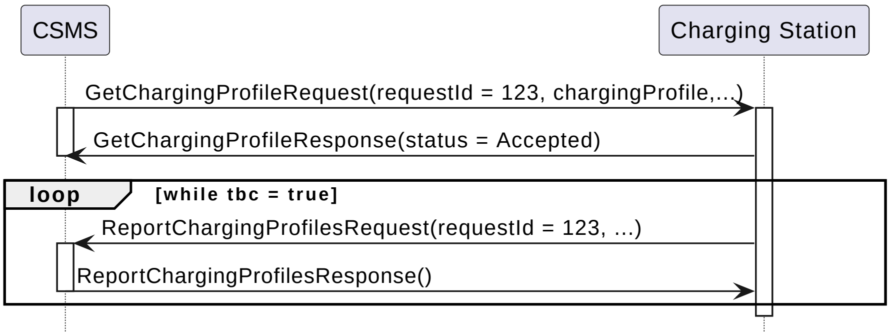

CSMS
Charging Station
GetChargingProfileRequest(requestId = 123, chargingProfile,...)
GetChargingProfileResponse(status = Accepted)
loop
[while tbc = true]
ReportChargingProfilesRequest(requestId = 123, ...)
ReportChargingProfilesResponse()

Figure 130. Sequence diagram of the use case "Get Charging Profiles"

K09 - Get Charging Profiles - Requirements

Table 105. K09 - Requirements

ID	Precondition	Requirements	Note
K09.FR.01	When *requestId* is set in the GetChargingProfilesRequest	The Charging Station SHALL set the *requestId* 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 *tbc* flag to **true** for all ReportChargingProfilesRequest messages except the last.	
K09.FR.03		The CSMS SHALL specify in *chargingProfile* criteria in GetChargingProfilesRequest either: - a (list of) *chargingProfileId(s)* OR - one or more of the fields *stackLevel*, *chargingLimitSource*, *chargingProfilePurpose*.	These fields are filter values of equal importance, but because a *chargingProfileId* uniquely identifies a charging profile, the other fields are not needed if *chargingProfileIds* are used.
K09.FR.04	If *evseId* 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 *chargingProfile*.	
K09.FR.05	If *evseId* 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 *chargingProfile*.	EVSE #0 can have a `ChargingStationMaxProfile`, `ChargingStationExternalConstraints` or a `TxDefaultProfile`. Note, that a `TxDefaultProfile` is not applied to EVSE #0 but to all individual EVSEs (see K01.FR.14).
K09.FR.06	If *evseId* is NOT set in the GetChargingProfilesRequest	The Charging Station SHALL report all installed charging profiles that match all fields in *chargingProfile*.	

K10 - Clear Charging Profile

No.	Type	Description
1	**Name**	Clear Charging Profile

No.	Type	Description
2	**ID**	K10
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 *Successfully* cleared. **Failure postcondition:** The requested charging profiles are *not* cleared, as no ChargingProfile is found.
7	**Error handling**	n/a
8	**Remark(s)**	n/a

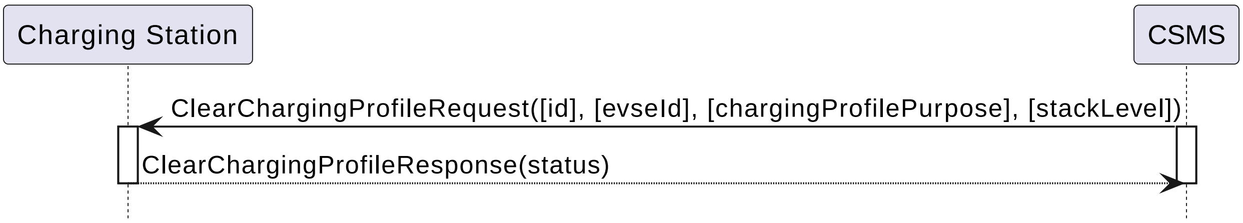

Charging Station
CSMS
ClearChargingProfileRequest([id], [evseId], [chargingProfilePurpose], [stackLevel])
ClearChargingProfileResponse(status)

Figure 131. Sequence Diagram of the use case "Clear Charging Profile"

K10 - Clear Charging Profile - Requirements

Table 106. 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 *Unknown*.	
K10.FR.02		The CSMS SHALL either specify a chargingProfile.id OR include one or more of the fields stackLevel, evseId and chargingProfilePurpose in the ClearChargingProfileRequest to specify which Charging Profiles need to be cleared.	
K10.FR.03	Upon receipt of a ClearChargingProfileRequest with a specified *chargingProfileId* AND the chargingProfilePurpose of the referenced ChargingProfile is NOT `ChargingStationExternalConstraints`	The Charging Station SHALL clear the Charging Profile with the matching id and respond with a ClearChargingProfileResponse message with *status* = `Accepted`.	
K10.FR.04 **(2.1)**	NOT K10.FR.03 AND NOT K10.FR.08 AND Upon receipt of a ClearChargingProfileRequest, with optional values for *evseId*, *chargingProfilePurpose*, *stackLevel*	The Charging Station SHALL clear the ChargingProfile(s) that match (as logical AND) the values in the request, except those for that have ChargingProfilePurpose = `ChargingStationExternalConstraints` or `LocalGeneration` and respond with a ClearChargingProfileResponse message with *status* = `Accepted`.	
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.	

ID	Precondition	Requirement definition	Note
K10.FR.06		The CSMS SHALL NOT set chargingProfilePurpose to ChargingStationExternalConstraints in a ClearChargingProfileRequest.	
K10.FR.06 **(2.1)**		The CSMS SHALL NOT set chargingProfilePurpose to ChargingStationExternalConstraints or LocalGeneration 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 **(2.1)**	Upon receipt of a ClearChargingProfileRequest, with optional values for *evseId*, *chargingProfilePurpose*, *stackLevel* AND the matched ChargingProfile(s) all have ChargingProfilePurpose = `ChargingStationExternalConstraints` or `LocalGeneration`	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 **(2.1)**	Upon receipt of a ClearChargingProfileRequest with a specified *chargingProfileId* AND the chargingProfilePurpose of the referenced ChargingProfile = `ChargingStationExternalConstraints` or `LocalGeneration`	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

Updated in OCPP 2.1

No.	Type	Description
1	**Name**	Set / Update External Charging Limit With Ongoing Transaction
2	**ID**	K11
3	**Objective(s)**	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. The External Control System can be a DSO, but also a smart meter or a home energy management system. The interface between External Control System and Charging Station is not specified. It can be any protocol that is supported by Charging Station for this purpose, even OCPP.
	Actors	External Control System, Charging Station, CSMS
	Scenario description	**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. The external system can set/clear a charging limit/schedule on the Charging Station via a direct connection to the Charging Station.
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.
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. An example of an external system is given, in this case a smart meter that might set an external charging limit to protect against overloading the local grid connection, but this could be any other external system that sets a charging limit. It is up to the Charging Station implementation to decide whether to represent the external limits for `ChargingStationExternalConstraints` as an Absolute or Relative charging profile or as a Dynamic charging profile with an *operationMode* = `ExternalLimits`.

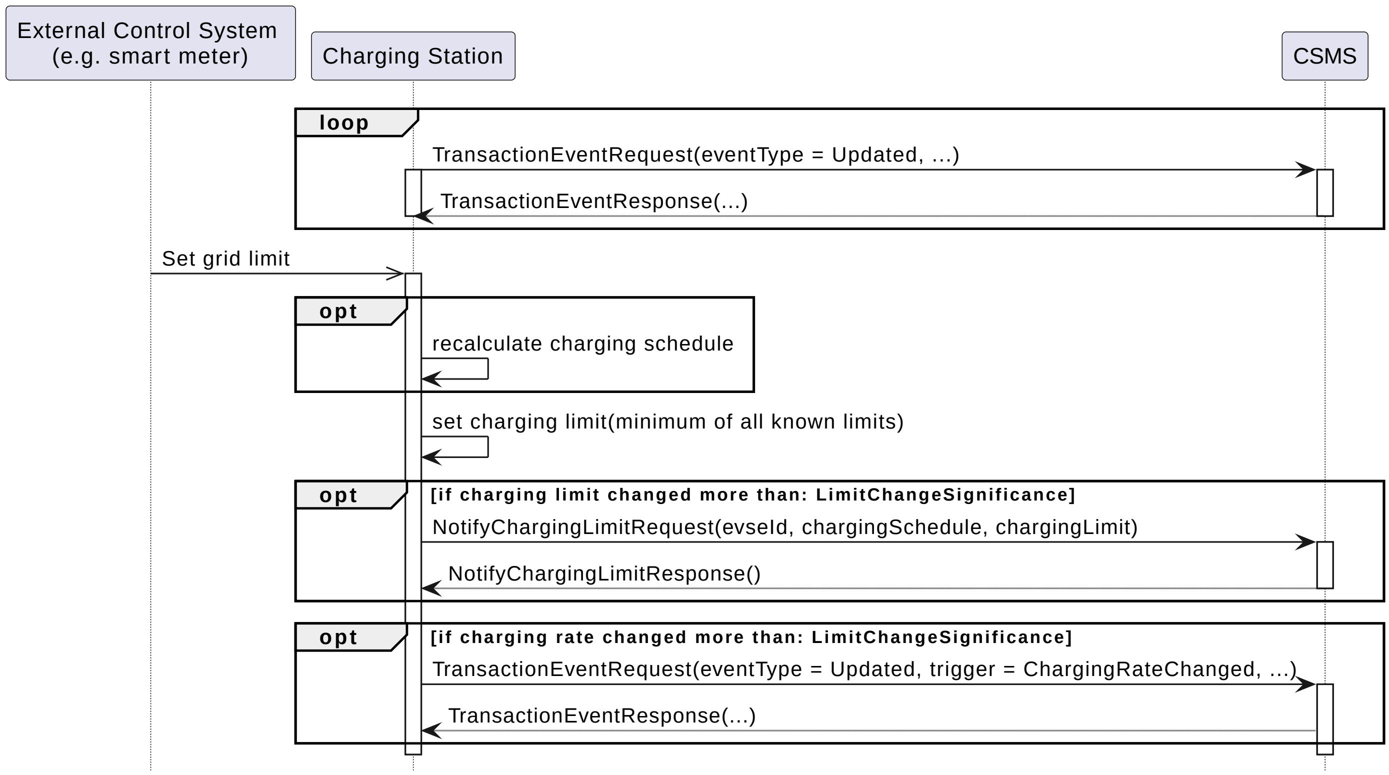

External Control System (e.g. smart meter)
Charging Station
CSMS
loop
TransactionEventRequest(eventType = Updated, ...)
TransactionEventResponse(...)
Set grid limit
opt
recalculate charging schedule
set charging limit(minimum of all known limits)
opt
[if charging limit changed more than: LimitChangeSignificance]
NotifyChargingLimitRequest(evseId, chargingSchedule, chargingLimit)
NotifyChargingLimitResponse()
opt
[if charging rate changed more than: LimitChangeSignificance]
TransactionEventRequest(eventType = Updated, trigger = ChargingRateChanged, ...)
TransactionEventResponse(...)

Figure 132. Sequence diagram of the use case "Setting / Updating External Charging Limit with Ongoing Transaction"

K11 - Set / Update External Charging Limit With Ongoing Transaction - Requirements

Table 107. K11 - Requirements

ID	Precondition	Requirements	Note
K11.FR.01 **(2.1)**	When an external charging limit/schedule is received during an ongoing transaction AND `ExternalConstraintsProfileDisallowed` is false or absent	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.	
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.	

ID	Precondition	Requirements	Note
K11.FR.06 **(2.1)**	K11.FR.01	The Charging Station SHALL use purpose `ChargingStationExternalConstraints` when reporting about this limit (i.e. 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. See use case K29 for the use of `Dynamic` charging profiles and external limits.
K11.FR.07 **(2.1)**	When an external charging limit/schedule is received during an ongoing transaction AND `ExternalConstraintsProfileDisallowed` is true	The Charging Station SHALL ignore the external charging limit/schedule.	CSMS will need to set a charging profile with *operationMode* = `ExternalLimits` to allow this. See use case K29.

K12 - Set / Update External Charging Limit Without Ongoing Transaction

Updated in OCPP 2.1

No.	Type	Description
1	**Name**	Set / Update External Charging Limit Without Ongoing Transaction
2	**ID**	K12
3	**Objective(s)**	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**	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. The External Control System can be a DSO, but also a smart meter or a home energy management system. The interface between External Control System and Charging Station is not specified. It can be any protocol that is supported by Charging Station for this purpose, even OCPP.
	Actors	External Control System, Charging Station, CSMS
	Scenario description	**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. The external system can set/clear a charging limit/schedule on the Charging Station via a direct connection to the Charging Station.
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.
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. An example of an external system is given, in this case a smart meter that might set an external charging limit to protect against overloading the local grid connection, but this could be any other external system that sets a charging limit. It is up to the Charging Station implementation to decide whether to represent the external limits for `ChargingStationExternalConstraints` as an Absolute or Relative charging profile or as a Dynamic charging profile with an *operationMode* = `ExternalLimits`.

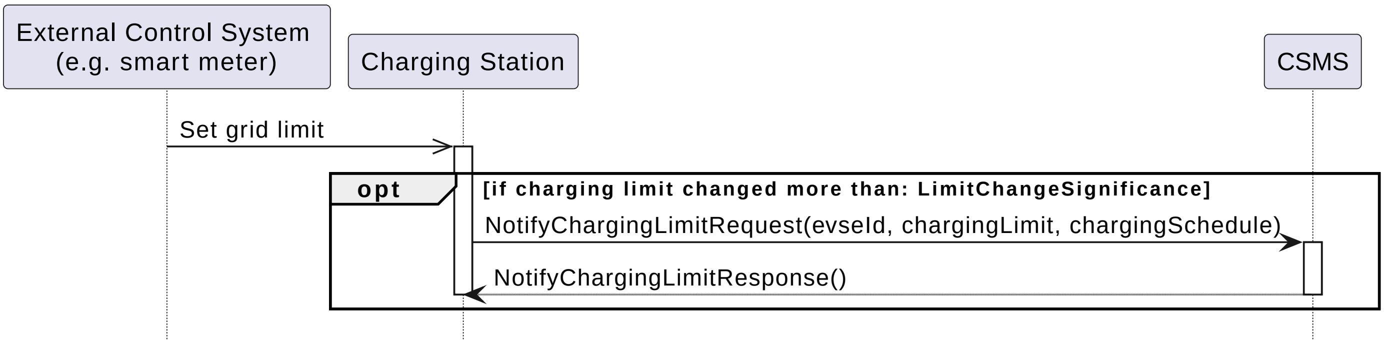

External Control System
(e.g. smart meter)
Charging Station
CSMS
Set grid limit
opt
[if charging limit changed more than: LimitChangeSignificance]
NotifyChargingLimitRequest(evseId, chargingLimit, chargingSchedule)
NotifyChargingLimitResponse()

Figure 133. Sequence diagram of the use case "Set / Update External Charging Limit Without Ongoing Transaction"

K12 - Set / Update External Charging Limit Without Ongoing Transaction - Requirements

Table 108. 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.	Same as K11.FR.02
K12.FR.03	K12.FR.02 AND `EnableNotifyChargingLimitWithSchedules` is true	The NotifyChargingLimitRequest SHALL contain the charging limit/schedule as set by the external system.	Same as K11.FR.03
K12.FR.04	K12.FR.02	The Charging Station SHALL NOT set the chargingLimitSource to CSO in the NotifyChargingLimitRequest.	Same as K11.FR.05
K12.FR.05	When an external charging limit/schedule is received	The Charging Station SHALL use purpose ChargingStationExternalConstraints when reporting about this limit (i.e. 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. See use case K29 for the use of `Dynamic` charging profiles and external limits. Same as K11.FR.06.

K13 - Reset / Release External Charging Limit

No.	Type	Description
1	**Name**	Reset / Release External Charging Limit
2	**ID**	K13
3	**Objective(s)**	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

No.	Type	Description
	Scenario description	**1.** External control system releases/removes a charging limit/schedule on the Charging Station **2.** When a transaction is ongoing, the Charging Station calculates the new Charging Schedule and adjusts charging speed. **3.** The Charging Station sends a ClearedChargingLimitRequest to notify the CSMS. **4.** The CSMS acknowledges with a ClearedChargingLimitResponse to the Charging Station. **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. **6.** The CSMS acknowledges with a TransactionEventResponse to the Charging Station.
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.
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.

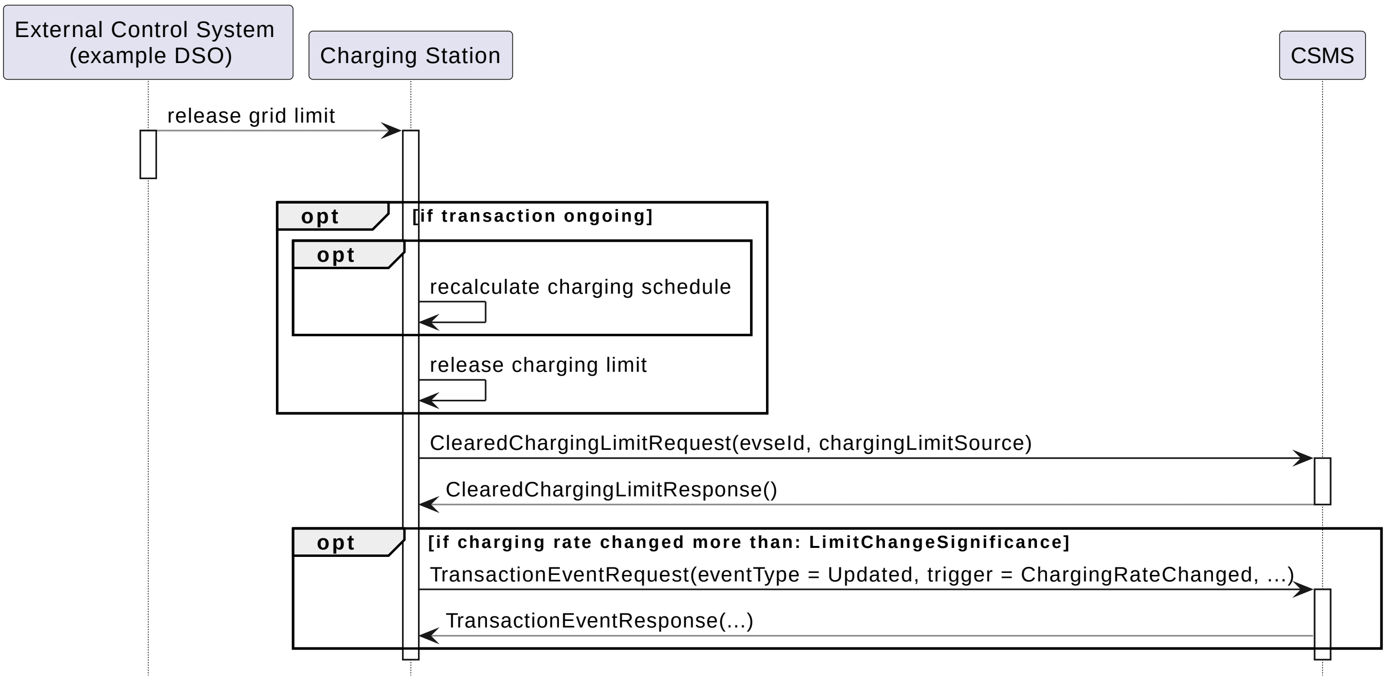

External Control System (example DSO)
Charging Station
CSMS
release grid limit
opt
[if transaction ongoing]
opt
recalculate charging schedule
release charging limit
ClearedChargingLimitRequest(evseId, chargingLimitSource)
ClearedChargingLimitResponse()
opt
[if charging rate changed more than: LimitChangeSignificance]
TransactionEventRequest(eventType = Updated, trigger = ChargingRateChanged, ...)
TransactionEventResponse(...)

Figure 134. Sequence diagram of the use case "Release / Reset External Charging Limit"

K13 - Reset / Release External Charging Limit - Requirements

Table 109. K13 - Requirements

ID	Precondition	Requirements
K13.FR.01	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.
K13.FR.03	K13.FR.01 AND A transaction is ongoing 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

No.	Type	Description
1	**Name**	Handle external charging limit with a local controller
2	**ID**	K14
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	**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 releases a charging limit/schedule to Local Controller. **6.** Local Controller sends a ClearedChargingLimitRequest message to the CSMS. **7.** Local Controller clears 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. Configuration variable `ExternalControlSignalsEnabled` = true.
6	**Postcondition(s)**	**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. **Failure postcondition:** The CSMS is not informed about the changed charging limit. The External Control System is not able to change the charging limit.
7	**Error handling**	n/a
8	**Remark(s)**	n/a

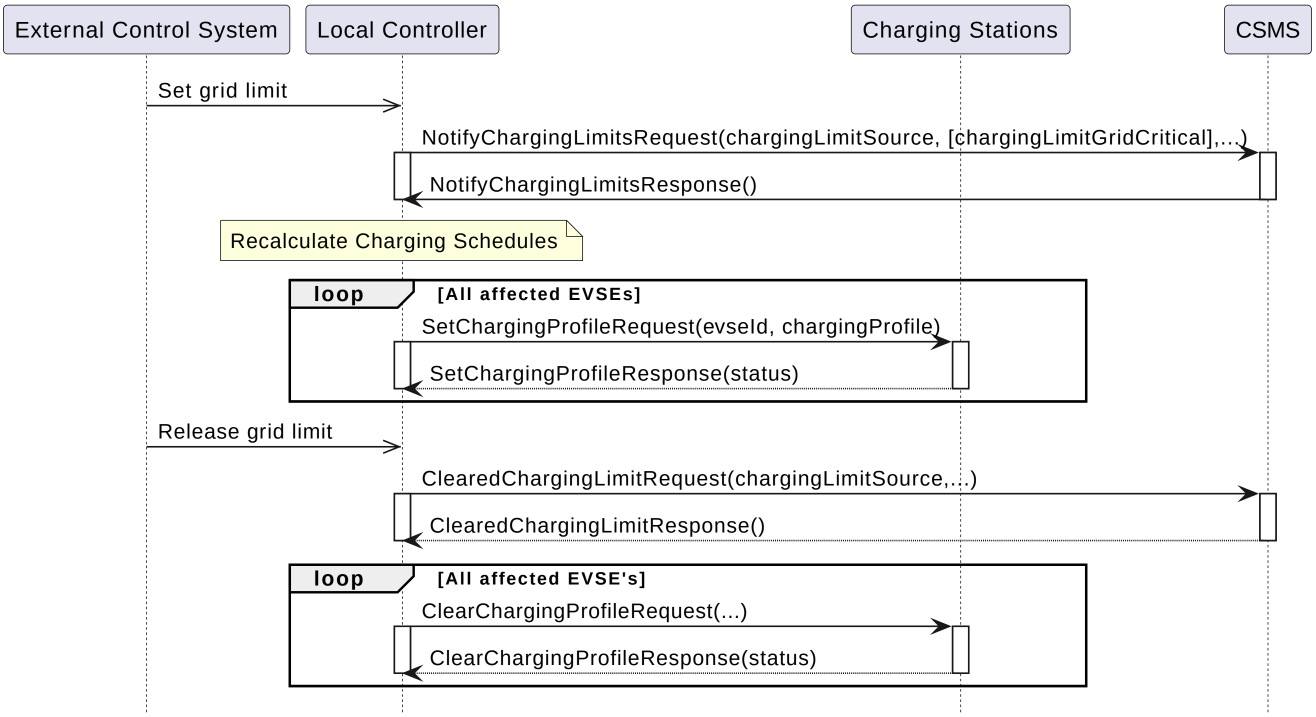

External Control System
Local Controller
Charging Stations
CSMS
Set grid limit
NotifyChargingLimitsRequest(chargingLimitSource, [chargingLimitGridCritical],...)
NotifyChargingLimitsResponse()
Recalculate Charging Schedules
loop
[All affected EVSEs]
SetChargingProfileRequest(evseId, chargingProfile)
SetChargingProfileResponse(status)
Release grid limit
ClearedChargingLimitRequest(chargingLimitSource,...)
ClearedChargingLimitResponse()
loop
[All affected EVSE's]
ClearChargingProfileRequest(...)
ClearChargingProfileResponse(status)

Figure 135. Sequence Diagram: External Charging Limit with Local Controller.

K14 - External Charging Limit with Local Controller - Requirements

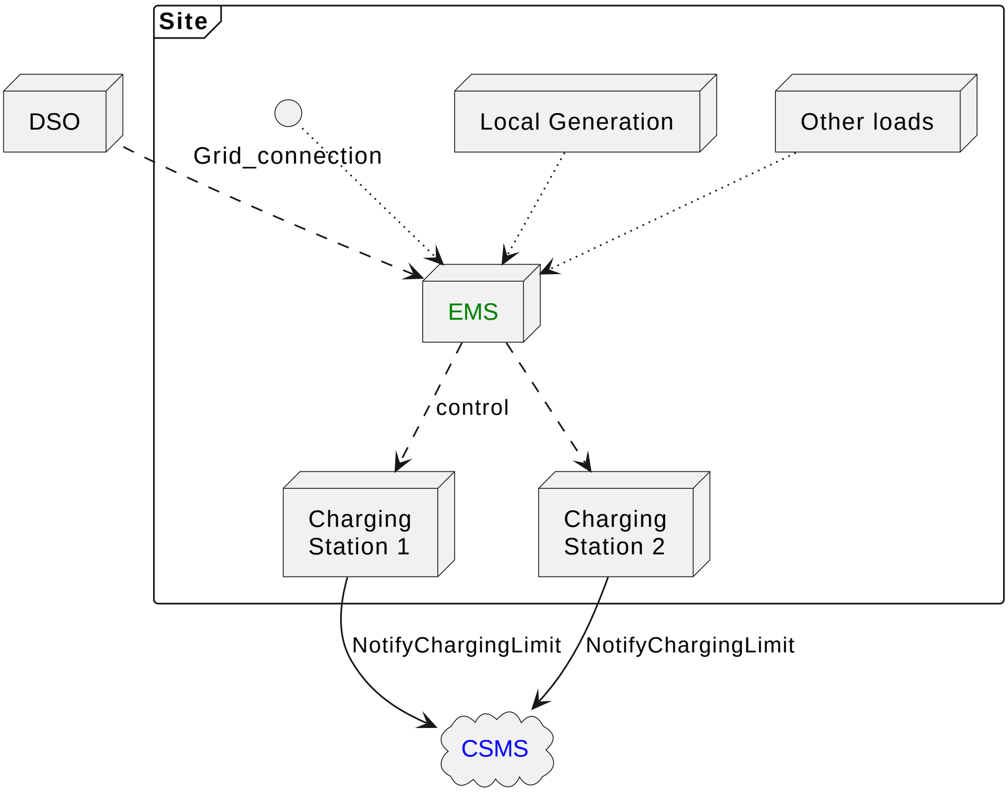
Table 110. K14 - Requirements

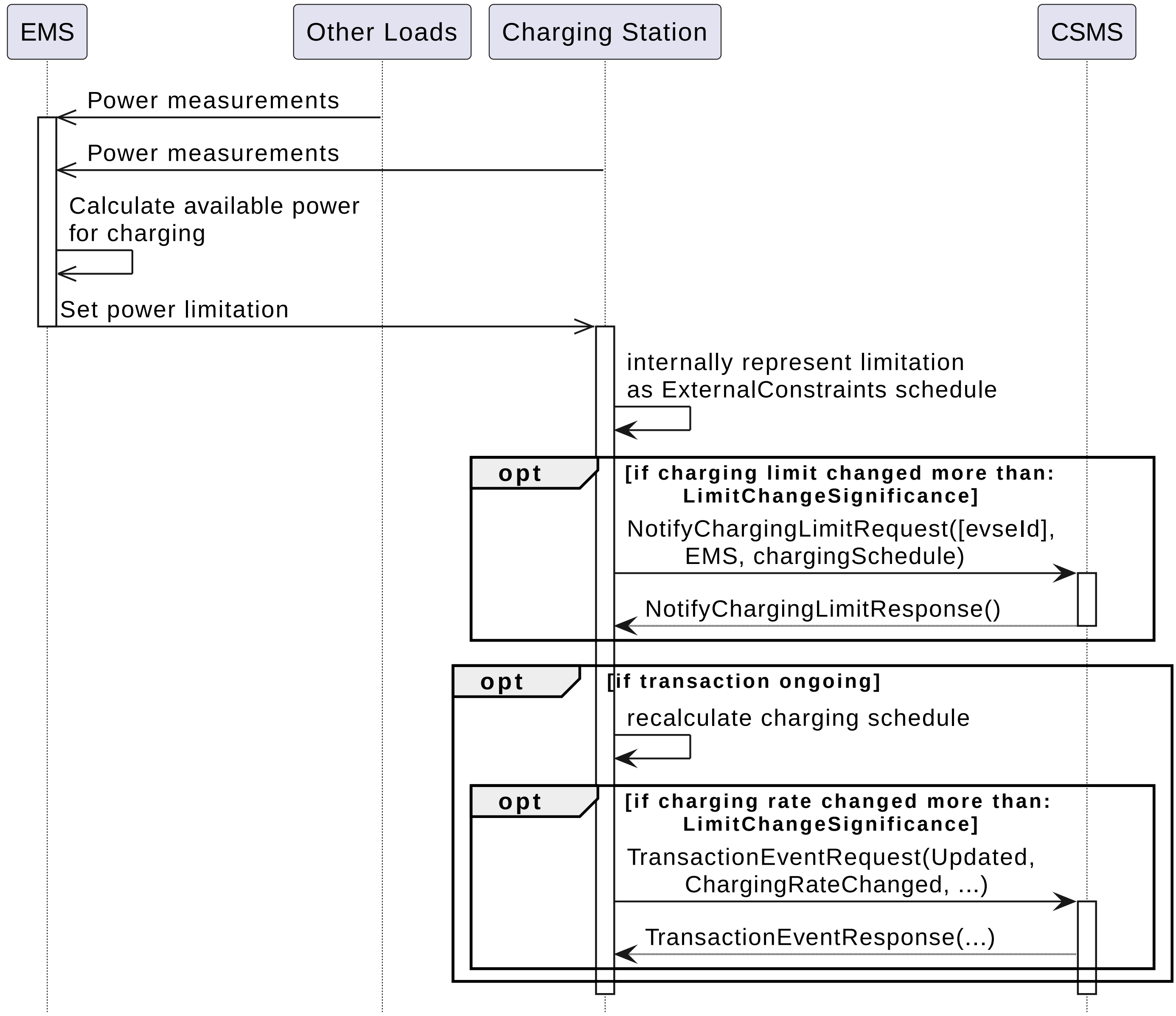
ID	Precondition	Requirement definition
K14.FR.01	When an external charging limit/schedule is received	The total load of all Charging Stations SHALL NOT exceed this given limit.
K14.FR.02	K14.FR.01 AND Charging limit changed by more than: `LimitChangeSignificance`	The Local Controller SHALL inform the 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.

K23 - Smart Charging with EMS connected to Charging Stations

New in OCPP 2.1

No.	Type	Description
1	**Name**	Smart Charging with EMS connected to Charging Stations
2	**ID**	K23 *(see also K11, K12)*
3	**Objective(s)**	To describe how smart charging can be performed by an EMS that has a direct connection to a Charging Station. The setting of the charging limits is described in detail in use cases K11 and K12 for any External Control System. This use case describes the specific situation where the External Control System is an EMS in a building.

No.	Type	Description
4	**Description**	EMS is managing the load within a site. It regularly updates current or power limitations for the Charging Stations based on other loads within the site, such that the local grid connection is not overloaded.  Site DSO Grid_connection Local Generation Other loads EMS control Charging Station 1 Charging Station 2 NotifyChargingLimit NotifyChargingLimit CSMS
	Actors	EMS, Charging Station, CSMS
	Scenario description	1. EMS measures power consumption from other loads in the building 2. EMS determines available power for Charging Stations based on the capacity of the local grid connection, other loads and the amount of locally generated energy (e.g. PV panels) 3. EMS sets power (or current) limit to Charging Stations via a direct connection. (The protocol between EMS and Charging Station is not relevant for the use case). 4. Charging Station internally represents the given limitation as a charging profile with *chargingProfilePurpose* = `ChargingStationExternalConstraints`. (Depending on EMS this can be a single limit value or a schedule of limits over time). 5. Charging Station notifies CSMS of the received limit if it changed by more than the configured threshold: `LimitChangeSignificance`, by sending a NotifyChargingLimitRequest message with *chargingLimitSource* = EMS and (optionally) a *chargingSchedule* that represents the limit(s). 6. If a transaction is ongoing, Charging Station will send a TransactionEventRequest message with *trigger* = `ChargingRateChanged` to record that a power limitation occurred during the transaction.
5	**Prerequisites**	EMS is able to control power level of Charging Stations. Configuration variable `ExternalControlSignalsEnabled` = true.
6	**Post conditions**	Charging power is reduced when needed to avoid overloading local grid connection.
7	**Error Handling**	
8	**Remarks**	If the other loads in the building consume less than the amount of locally generated energy, then EMS might be able to let the Charging Stations use this as additional capacity on top of the capacity of the local grid connection. This situation is described in K27 - Smart Charging with EMS and LocalGeneration.


EMS
Other Loads
Charging Station
CSMS
Power measurements
Power measurements
Calculate available power for charging
Set power limitation
internally represent limitation as ExternalConstraints schedule
opt
[if charging limit changed more than: LimitChangeSignificance]
NotifyChargingLimitRequest([evseId], EMS, chargingSchedule)
NotifyChargingLimitResponse()
opt
[if transaction ongoing]
recalculate charging schedule
opt
[if charging rate changed more than: LimitChangeSignificance]
TransactionEventRequest(Updated, ChargingRateChanged, ...)
TransactionEventResponse(...)

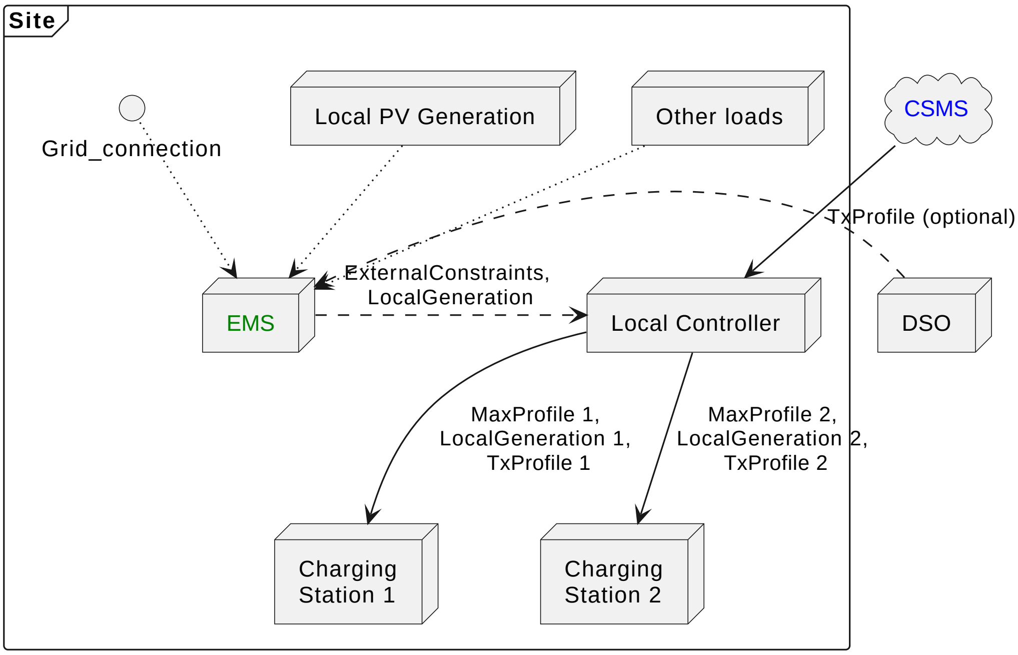
K23 - Smart Charging with EMS connected to Charging Stations - Requirements

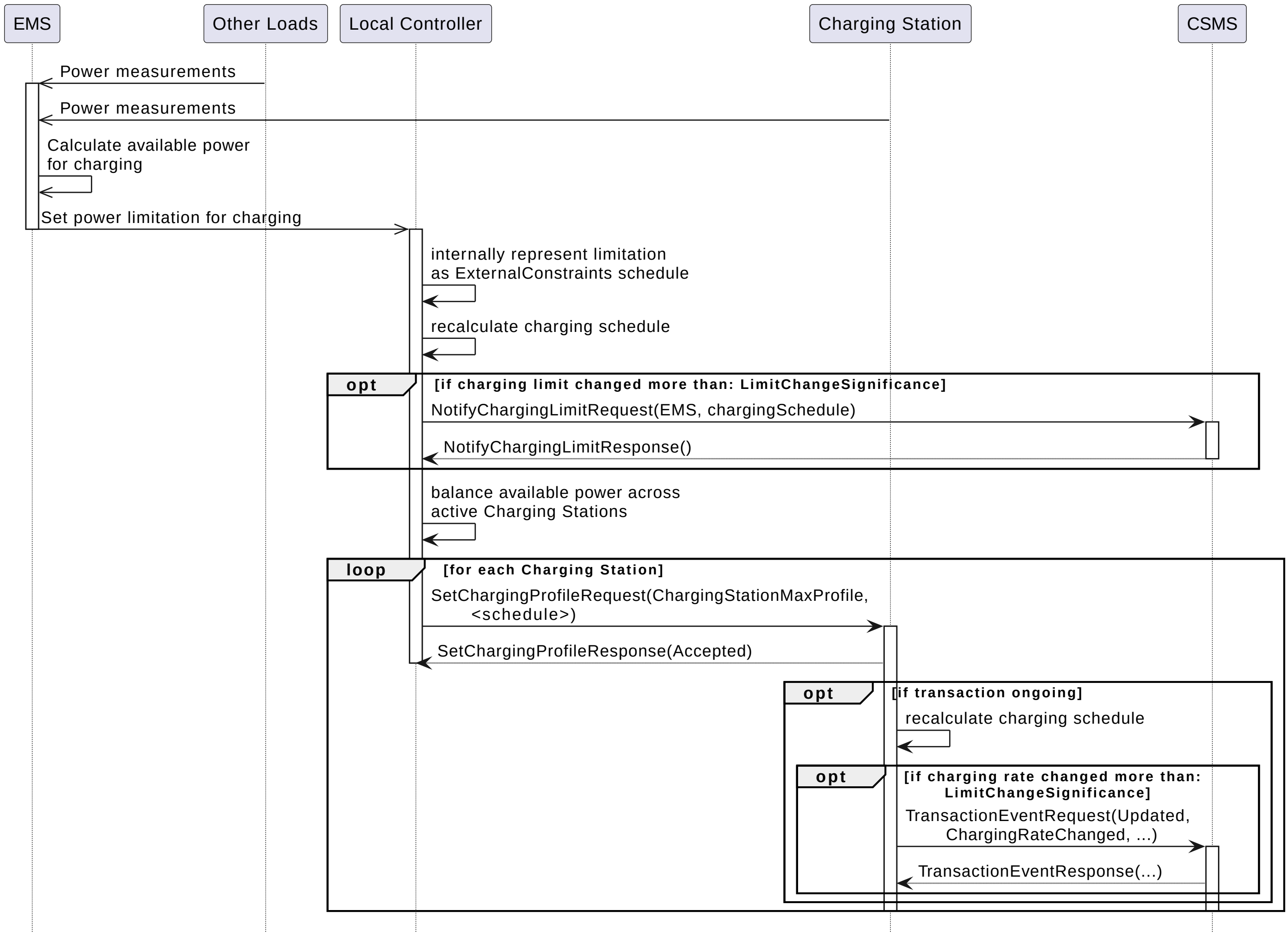
The requirements for this use case are already covered in use cases K11 and K12.

K24 - Smart Charging with EMS connected to Local Controller

New in OCPP 2.1

No.	Type	Description
1	**Name**	Smart Charging with EMS connected to Local Controller
2	**ID**	K24 *(see also K14)*
3	**Objective(s)**	To describe how smart charging can be performed by an EMS that is connected to a Local Controller that manages Charging Stations in the building. The setting of the charging limits to a Local Controller is described in detail in use case K14 for any External Control System. This use case describes the specific situation where the External Control System is an EMS in a building.

No.	Type	Description
4	**Description**	EMS is managing the load within a site. It regularly updates a current or power limitation for the cluster of Charging Stations that is managed by a Local Contoller, such that the local grid connection is not overloaded.  Site Grid_connection Local PV Generation Other loads CSMS TxProfile (optional) ExternalConstraints, LocalGeneration EMS Local Controller DSO MaxProfile 1, LocalGeneration 1, TxProfile 1 MaxProfile 2, LocalGeneration 2, TxProfile 2 Charging Station 1 Charging Station 2
	Actors	EMS, Local Controller, Charging Station, CSMS
	Scenario description	1. EMS measures power consumption from other loads in the building 2. EMS determines available power for EV charging based on the capacity of the local grid connection, other loads and the amount of locally generated energy (e.g. PV panels) 3. EMS sets power (or current) limit to the Local Controller via a direct connection. (The protocol between EMS and Local Controller is not relevant for the use case). 4. Local Controller internally represents the given limitation as charging profile with *chargingProfilePurpose* = `ChargingStationExternalConstraints`. (Depending on EMS this can be a single limit value or a schedule of limits over time). 5. Local Controller notifies CSMS of the received limit if it changed by more than the configured threshold: `LimitChangeSignificance`, by sending a NotifyChargingLimitRequest message with *chargingLimitSource* = EMS and (optionally) a *chargingSchedule* that represents the limit(s). 6. Local Controller balances the available current or power across all connected Charging Stations by sending each Charging Station an (updated) charging profile via a SetChargingProfileRequest message with *chargingProfilePurpose* = `ChargingStationMaxProfile`. 7. If a transaction is ongoing, and the received charging profile limits have changed more than `LimitChangeSignificance`, Charging Station will send a TransactionEventRequest message with *trigger* = `ChargingRateChanged` to record that a power limitation occurred during the transaction.
5	**Prerequisites**	Local Controller is able to load-balance across Charging Stations at site.
6	**Post conditions**	Charging power is reduced when needed to avoid overloading local grid connection.
7	**Error Handling**	
8	**Remarks**	A Local Controller acts as a local CSMS for the site. The *chargingProfilePurpose* must therefore not be `ChargingStationExternalConstraints`, because the Local Controller is not seen as an External Control System. A NotifyChargingLimitRequest is only sent by the Local Controller (not by the individual Charging Stations), since that is the component that receives the external charging limit.


EMS
Other Loads
Local Controller
Charging Station
CSMS
Power measurements
Power measurements
Calculate available power for charging
Set power limitation for charging
internally represent limitation as ExternalConstraints schedule
recalculate charging schedule
opt
[if charging limit changed more than: LimitChangeSignificance]
NotifyChargingLimitRequest(EMS, chargingSchedule)
NotifyChargingLimitResponse()
balance available power across active Charging Stations
loop
[for each Charging Station]
SetChargingProfileRequest(ChargingStationMaxProfile, <schedule>)
SetChargingProfileResponse(Accepted)
opt
[if transaction ongoing]
recalculate charging schedule
opt
[if charging rate changed more than: LimitChangeSignificance]
TransactionEventRequest(Updated, ChargingRateChanged, ...)
TransactionEventResponse(...)

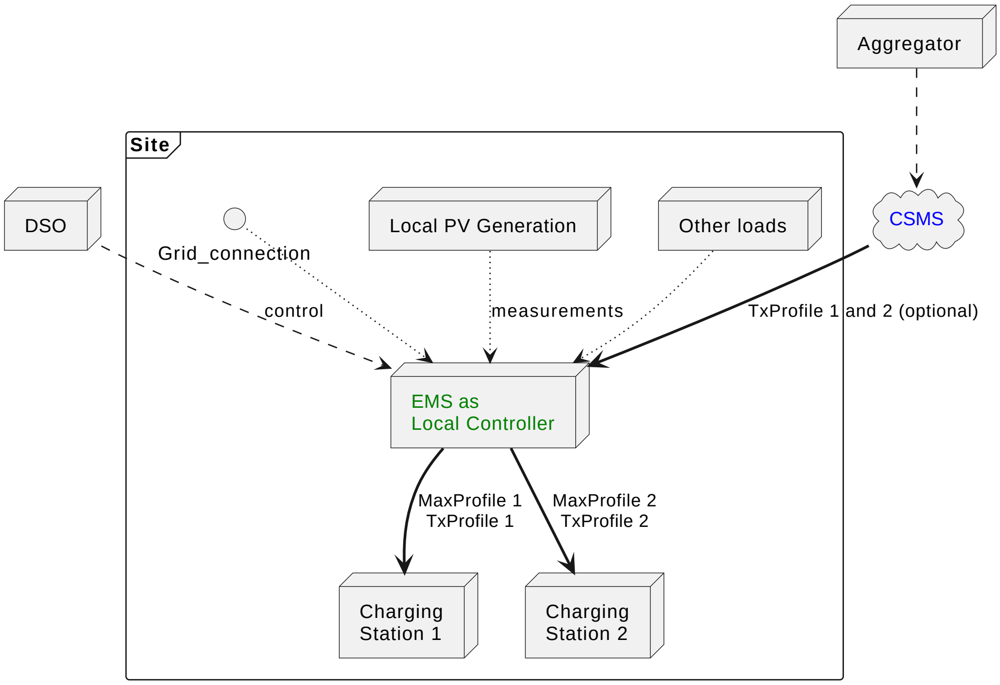
K24 - Smart Charging with EMS connected to Local Controller - Requirements

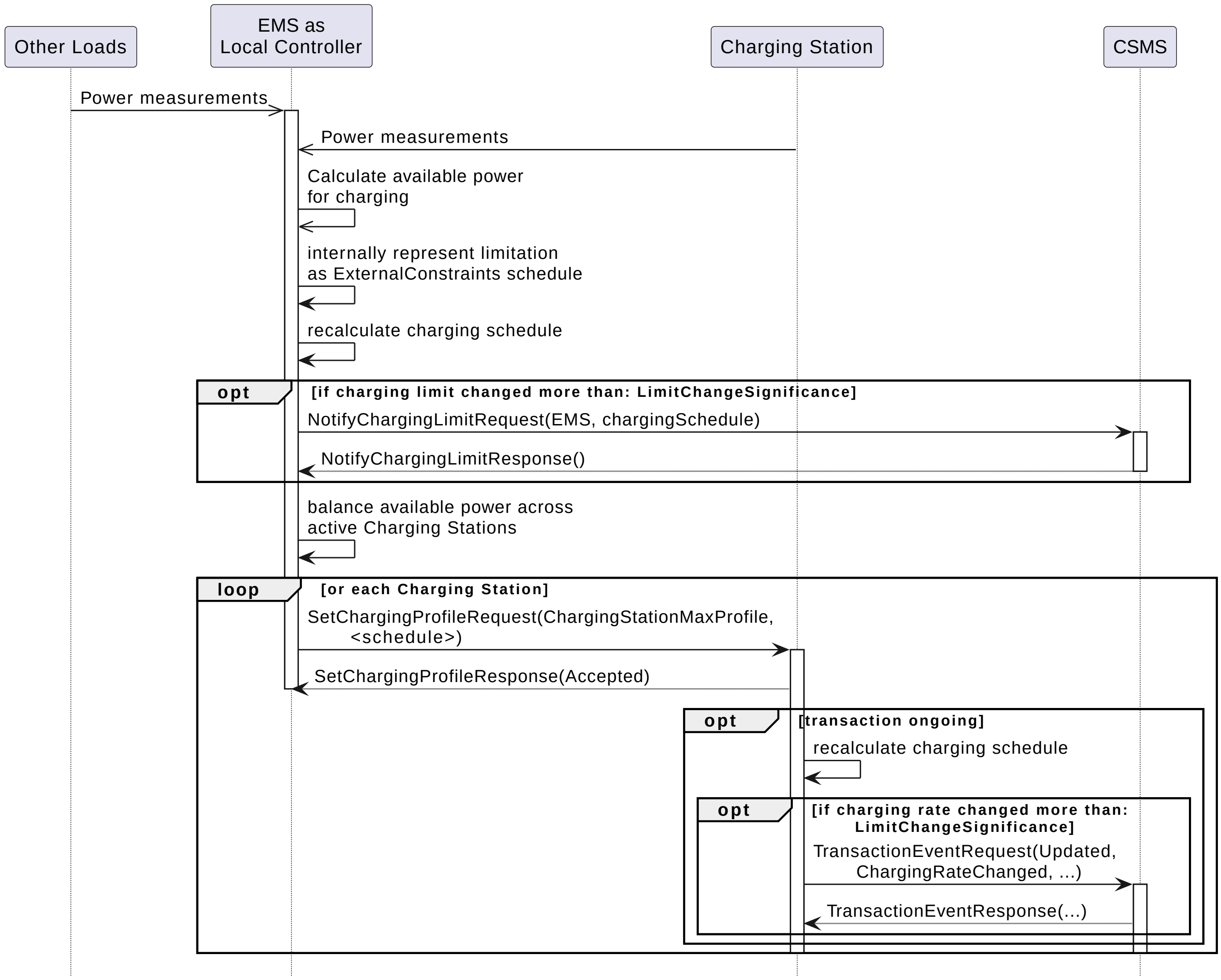
The requirements for this use case are already covered in use case K14.

K25 - Smart Charging with EMS acting as a Local Controller

New in OCPP 2.1

No.	Type	Description
1	**Name**	Smart Charging with EMS acting as a Local Controller
2	**ID**	K25 *(see also K14)*
3	**Objective(s)**	To describe how smart charging can be performed by an EMS that is acting as a Local Controller that manages Charging Stations in the building. The setting of the charging limits to a Local Controller is described in detail in use case K14 for any External Control System. This use case describes the specific situation where the functionality of EMS and Local Controller are combined in a single component.

No.	Type	Description
4	**Description**	EMS is managing the load within a site and also acting as a Local Controller for Charging Stations. It controls charging profiles for the Charging Stations on site, such that the local grid connection is not overloaded. More advanced scheduling is possible, since this component is a Local Controller and all OCPP traffic passes through it, which allows the EMS-part of this component to be aware of the state of ongoing transactions, such as charging schedules imposed by CSMS, state of charge of the EV, and planned time of departure.  Aggregator Site DSO Grid_connection Local PV Generation Other loads CSMS control measurements TxProfile 1 and 2 (optional) EMS as Local Controller MaxProfile 1 TxProfile 1 MaxProfile 2 TxProfile 2 Charging Station 1 Charging Station 2
	Actors	EMS, Charging Station, CSMS
	Scenario description	1. EMS measures power consumption from other loads in the building. 2. EMS optionally collects data from OCPP traffic about ongoing transactions, state of charge and planned time of departure, when available. 3. EMS calculates a power profile for each Charging Station based on the state of charging transactions, other loads, the capacity of the local grid connection and the amount of locally generated energy (e.g. PV panels) 4. EMS (as Local Controller) sends each power profile as a charging profile of *chargingProfilePurpose* = `ChargingStationMaxProfile` to the Charging Stations, in addition to charging profiles (if any) from CSMS towards Charging Stations. 5. EMS (as Local Controller) notifies CSMS of a power limit if it changed by more than the configured threshold: `LimitChangeSignificance`, by sending a NotifyChargingLimitRequest message with *chargingLimitSource* = EMS and (optionally) a *chargingSchedule* that represents the limit(s). 6. If a transaction is ongoing, and the received charging profile limits have changed more than `LimitChangeSignificance`, Charging Station will send a TransactionEventRequest message with *trigger* = `ChargingRateChanged` to record that a power limitation occurred during the transaction.
5	**Prerequisites**	EMS and Local Controller are integrated and Local Controller part has load-balancing capabilities.
6	**Post conditions**	Charging power is reduced when needed to avoid overloading local grid connection.
7	**Error Handling**	
8	**Remarks**	A Local Controller acts as a local CSMS for the site. The *chargingProfilePurpose* must therefore not be `ChargingStationExternalConstraints`, because the Local Controller is not seen as an External Control System. A NotifyChargingLimitRequest is only sent by the Local Controller (not by the individual Charging Stations), since that is the component that receives the external charging limit.


Other Loads
EMS as Local Controller
Charging Station
CSMS
Power measurements
Power measurements
Calculate available power for charging
internally represent limitation as ExternalConstraints schedule
recalculate charging schedule
opt
[if charging limit changed more than: LimitChangeSignificance]
NotifyChargingLimitRequest(EMS, chargingSchedule)
NotifyChargingLimitResponse()
balance available power across active Charging Stations
loop
[or each Charging Station]
SetChargingProfileRequest(ChargingStationMaxProfile, <schedule>)
SetChargingProfileResponse(Accepted)
opt
[transaction ongoing]
recalculate charging schedule
opt
[if charging rate changed more than: LimitChangeSignificance]
TransactionEventRequest(Updated, ChargingRateChanged, ...)
TransactionEventResponse(...)

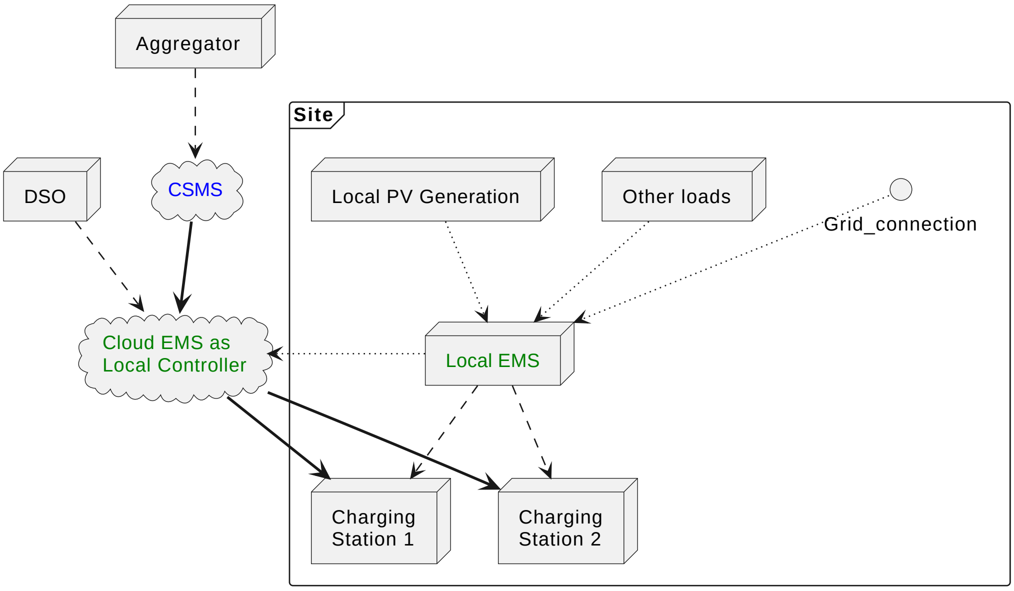
K25 - Smart Charging with EMS acting as a Local Controller - Requirements

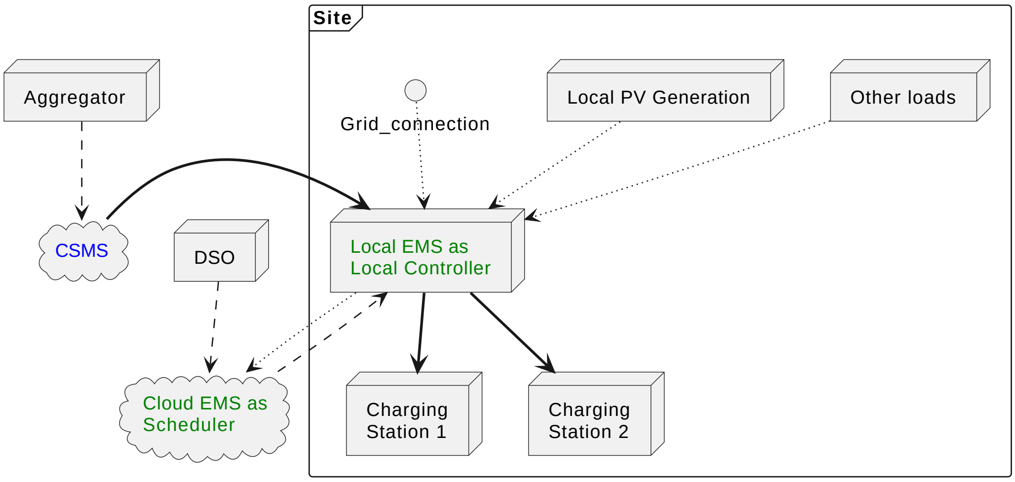
The requirements for this use case are already covered in use case K14.

K26 - Smart Charging with Hybrid Local & Cloud EMS

New in OCPP 2.1

No.	Type	Description
1	**Name**	Smart Charging with Hybrid Local & Cloud EMS
2	**ID**	K26 *(see also K14)*
3	**Objective(s)**	To describe how smart charging can be performed by an EMS in the cloud that is acting as a Local Controller that manages Charging Stations in the building. This use case is similar to K25. The difference is, that in this case EMS has cloud component and a local component.
4	**Description**	In this use case EMS exists in both a cloud and a local instance. It has two alternative topologies: 1. The part running in the cloud is responsible for scheduling and Local Controller functionality, and the other part running locally on site measures local loads and acts as a fail-safe protection against overloading the local grid connection. 2. The part running in the cloud is only responsible for scheduling, and the other part running locally on site acts as a Local Controller and measures local loads and acts as a fail-safe protection against overloading the local grid connection.
	Actors	Cloud EMS, Local EMS, Charging Station, CSMS

No.	Type	Description
	Scenario description	*Cloud EMS as Local Controller* EMS in the cloud can easily connect to external sources of information, such as weather forecasts and inputs from a DSO or TSO. Since this component is also a Local Controller and all OCPP traffic passes through it, this allows the EMS-part of this component to be aware of the state of ongoing transactions, such as charging schedules imposed by CSMS, state of charge of the EV, and planned time of departure. Together with information of external sources, this enables advanced scheduling.  Aggregator Site DSO CSMS Local PV Generation Other loads Grid_connection Cloud EMS as Local Controller Local EMS Charging Station 1 Charging Station 2
		1. Local EMS measures power consumption from other loads in the building, and passes this on to Cloud EMS. 2. Cloud EMS collects data from external inputs, like weather forecasts and DSO/TSO input. 3. Cloud EMS (as Local Controller) optionally collects data from OCPP traffic about ongoing transactions, state of charge and planned time of departure, when available. 4. Cloud EMS calculates a power profile for each Charging Station based on the state of charging transactions and information received from Local EMS, such as local loads, capacity of the local grid connection and the amount of locally generated energy (e.g. PV panels) 5. Cloud EMS (as Local Controller) sends each power profile as a charging profile of *chargingProfilePurpose* = `ChargingStationMaxProfile` to the Charging Stations, in addition to charging profiles (if any) from CSMS towards Charging Stations. 6. Cloud EMS (as Local Controller) notifies CSMS of a power limit if it changed by more than the configured threshold: `LimitChangeSignificance`, by sending a NotifyChargingLimitRequest message with *chargingLimitSource* = EMS and (optionally) a *chargingSchedule* that represents the limit(s). 7. If a transaction is ongoing, and the received charging profile limits have changed more than `LimitChangeSignificance`, Charging Station will send a TransactionEventRequest message with *trigger* = `ChargingRateChanged` to record that a power limitation occurred during the transaction.

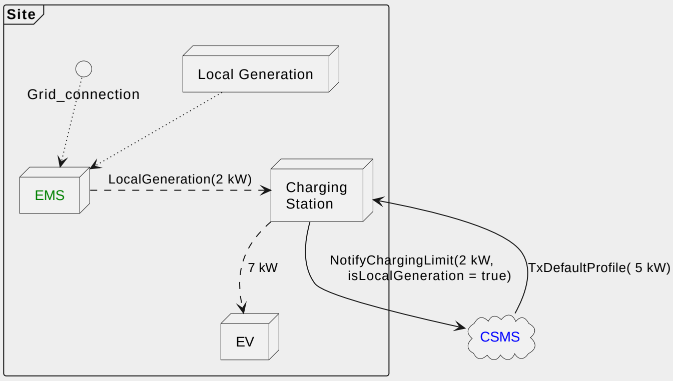
No.	Type	Description
	Scenario description #2	*Local EMS as Local Contoller* The Local EMS acts as a Local Controller and all OCPP traffic passes through it. As such, it is aware of the state of ongoing transactions, such as charging schedules imposed by CSMS, state of charge of the EV, and planned time of departure. EMS in the cloud can easily connect to external sources of information, such as weather forecasts and inputs from a DSO or TSO, but since it is not acting as a Local Controller, it will not be aware of the state on ongoing transactions, unless this information is passed on between Local EMS and Cloud EMS.  Site Aggregator Grid_connection Local PV Generation Other loads CSMS DSO Local EMS as Local Controller Cloud EMS as Scheduler Charging Station 1 Charging Station 2
		1. Local EMS measures power consumption from other loads in the building, and passes this on to Cloud EMS. 2. Local EMS (as Local Controller) optionally collects data from OCPP traffic about ongoing transactions, state of charge and planned time of departure, when available, but since scheduling takes place in the Cloud EMS, this information cannot be taking into account, unless it Local EMS sends it to Cloud EMS. 3. Cloud EMS collects data from external inputs, like weather forecasts and DSO/TSO input. 4. Cloud EMS calculates a power profile for each Charging Station based on information received from Local EMS, such as local loads, capacity of the local grid connection and the amount of locally generated energy (e.g. PV panels) and optionally information about ongoing transactions. 5. Cloud EMS sends power profiles to Local EMS. 6. Local EMS (as Local Controller) sends each power profile as a charging profile of *chargingProfilePurpose* = `ChargingStationMaxProfile` to the Charging Stations, in addition to charging profiles (if any) from CSMS towards Charging Stations. 7. Local EMS (as Local Controller) notifies CSMS of a power limit if it changed by more than the configured threshold: `LimitChangeSignificance`, by sending a NotifyChargingLimitRequest message with *chargingLimitSource* = EMS and (optionally) a *chargingSchedule* that represents the limit(s). 8. If a transaction is ongoing, and the received charging profile limits have changed more than `LimitChangeSignificance`, Charging Station will send a TransactionEventRequest message with *trigger* = `ChargingRateChanged` to record that a power limitation occurred during the transaction.
5	**Prerequisites**	EMS and Local Controller are integrated and Local Controller part has load-balancing capability.
6	**Post conditions**	Charging power is reduced when needed to avoid overloading local grid connection.
7	**Error Handling**	
8	**Remarks**	Although the Local Controller functionality of EMS can either run in the cloud or locally, it is often more convenient to run it in the cloud, because a cloud instance has more processing power, can more easily be connected to a cloud CSMS and Charging Stations, and the interface between Cloud EMS and Local EMS is simpler, because no transaction data needs to be exchanged.

K26 - Smart Charging with Hybrid Local & Cloud EMS - Requirements

The requirements for this use case are already covered in use case K14.

K27 - Smart Charging with EMS and LocalGeneration

New in OCPP 2.1

No.	Type	Description
1	**Name**	Smart Charging with EMS and LocalGeneration
2	**ID**	K27
3	**Objective(s)**	To show how locally available capacity can be taken into account.
4	**Description**	In the situation where more power is available for the Charging Station due to the local power generation which is not known by CSMS, e.g. from solar panels, this extra capacity can be specified in a charging profile of type `LocalGeneration`. In this use case 2 kW is available from local solar panels on top of a 5 kW TxDefaultProfile.  Site Local Generation Grid_connection EMS LocalGeneration(2 kW) Charging Station 7 kW NotifyChargingLimit(2 kW, isLocalGeneration = true) TxDefaultProfile(5 kW) EV CSMS
	Actors	Charging Station, CSMS, EMS
	Scenario description	*Using a predicted local generation schedule* 1. Assume CSMS has set a ChargingProfile with *chargingProfilePurpose* = `TxDefaultProfile`, *chargingProfileKind* = `Relative` and a *chargingSchedule* with *chargingRateUnit* = `W` and 1 *chargingSchedulePeriod* that defines a *limit* = 5000. 2. An external energy management system provides a schedule for the local generation by means of a protocol. The schedule contains two periods: first period for 2 kW, second period for 1 kW. (Duration of the periods is required, but not relevant to this example.) Charging Station represents this input internally as a ChargingProfile with *chargingProfilePurpose* = `LocalGeneration`, *chargingProfileKind* = `Absolute` and a *chargingSchedule* with *chargingRateUnit* = `W` and 2 *chargingSchedulePeriods* that define a *limit* = 2000 and a *limit* = 1000. 3. First period: a. If the charging limit changed by more than: `LimitChangeSignificance`, the Charging Station sends a NotifyChargingLimitRequest message to CSMS with a *chargingLimit* that has *chargingLimitSource* = EMS and *isLocalGeneration* = true, and the local generation charging schedule from that moment onwards. b. Charging Station internally calculates a composite schedule that allows for a maximum power consumption of 7000 W. 4. Second period, the local generation drops to 1 kW: a. If the charging limit changed by more than: `LimitChangeSignificance`, the Charging Station sends a NotifyChargingLimitRequest message to CSMS with a *chargingLimit* that has *chargingLimitSource* = EMS and *isLocalGeneration* = true, and the local generation charging schedule from that moment onwards. b. Charging Station internally calculates a composite schedule that allows for a maximum power consumption of 6000 W.

No.	Type	Description
	Scenario description #2	*Using Dynamic charging schedule* 1. Assume CSMS has set a ChargingProfile with *chargingProfilePurpose* = `TxDefaultProfile`, *chargingProfileKind* = `Relative` and a *chargingSchedule* with *chargingRateUnit* = `W` and 1 *chargingSchedulePeriod* that defines a *limit* = 5000. 2. An external energy management system provides a (dynamic) value for the local generation by means of a protocol. Charging Station represents this input internally as a ChargingProfile with *chargingProfilePurpose* = `LocalGeneration`, *chargingProfileKind* = `Dynamic` and a *chargingSchedule* with *chargingRateUnit* = `W` and 1 *chargingSchedulePeriod* with *operationMode* = `ExternalLimits`. 3. The value for *limit* in the `LocalGeneration` charging profile is continuously updated with the value from the external system. a. If the charging limit changed by more than: `LimitChangeSignificance`, the Charging Station sends a NotifyChargingLimitRequest message to CSMS with a *chargingLimit* that has *chargingLimitSource* = EMS and *isLocalGeneration* = true, and the local generation charging schedule from that moment onwards. b. Charging Station internally calculates a composite schedule that allows for a maximum power consumption of 5000 W (from `TxDefaultProfile`) plus *limit* from `LocalGeneration`.
5	**Prerequisites**	EMS is able to measure (and possibly predict) the amount of local generation, and send it to Charging Station in a way that it can recognize it as local generation. Configuration variable `ExternalControlSignalsEnabled` = true.
6	**Post conditions**	Charging Station can use available energy from local generation in addition to the local grid capacity.
7	**Error Handling**	
8	**Remarks**	If the external system provides a limit via a protocol that is not OCPP, e.g. ModBus, then Charging Station can represent this as an Absolute charging profile, that is replaced when the limit changes, or as a Dynamic charging profile with a single charging schedule period with *operationMode* = `ExternalLimits` in which the *limit* is dynamically updated. It is up to the Charging Station implementation to decide whether to represent the external limits for `LocalGeneration` as an Absolute or as a Dynamic charging profile with an *operationMode* = `ExternalLimits`.

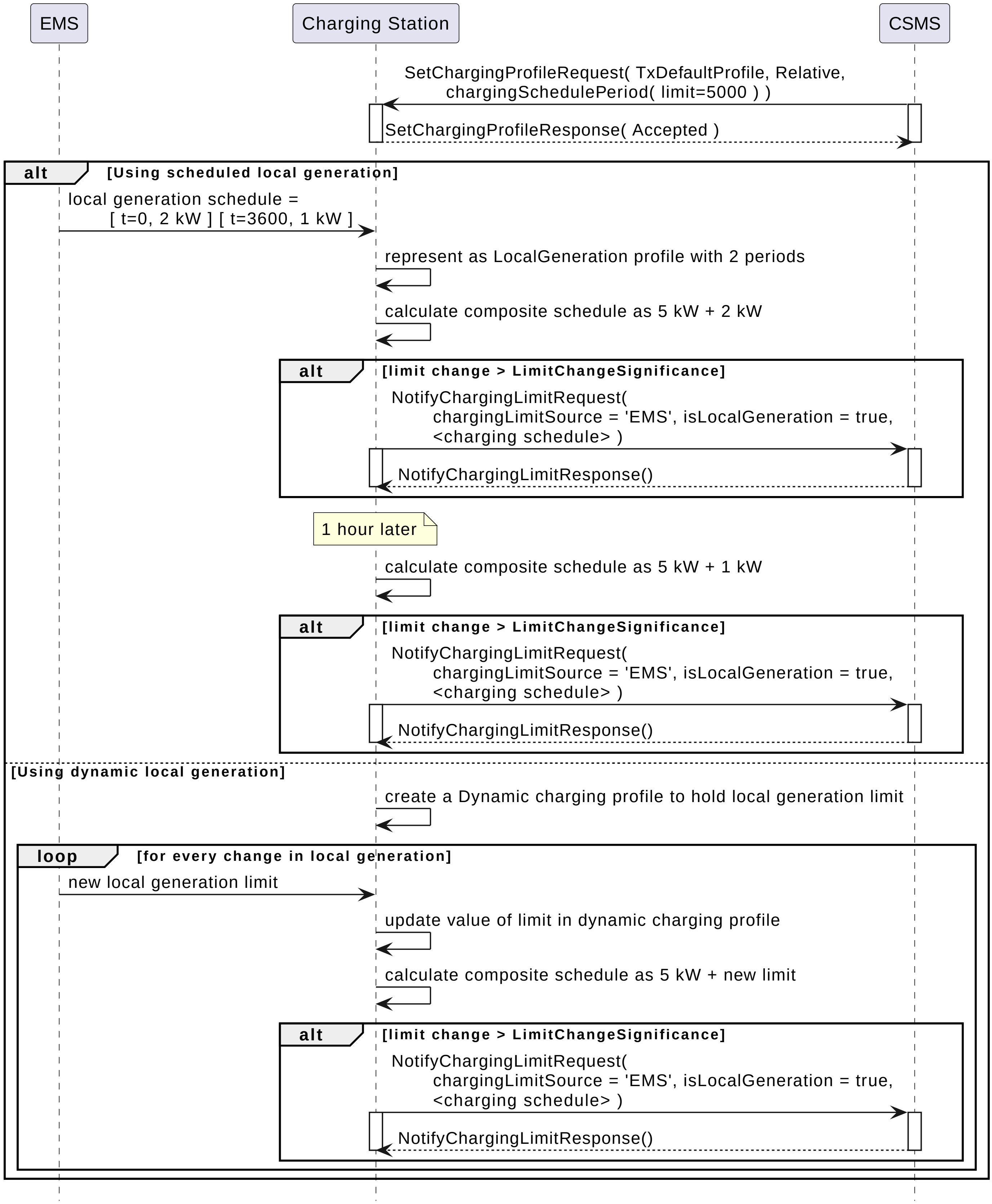

EMS
Charging Station
CSMS
SetChargingProfileRequest(TxDefaultProfile, Relative, chargingSchedulePeriod(limit=5000))
SetChargingProfileResponse(Accepted)
alt [Using scheduled local generation]
local generation schedule = [t=0, 2 kW] [t=3600, 1 kW]
represent as LocalGeneration profile with 2 periods
calculate composite schedule as 5 kW + 2 kW
alt [limit change > LimitChangeSignificance]
NotifyChargingLimitRequest(chargingLimitSource = 'EMS', isLocalGeneration = true, <charging schedule>)
NotifyChargingLimitResponse()
1 hour later
calculate composite schedule as 5 kW + 1 kW
alt [limit change > LimitChangeSignificance]
NotifyChargingLimitRequest(chargingLimitSource = 'EMS', isLocalGeneration = true, <charging schedule>)
NotifyChargingLimitResponse()
[Using dynamic local generation]
create a Dynamic charging profile to hold local generation limit
loop [for every change in local generation]
new local generation limit
update value of limit in dynamic charging profile
calculate composite schedule as 5 kW + new limit
alt [limit change > LimitChangeSignificance]
NotifyChargingLimitRequest(chargingLimitSource = 'EMS', isLocalGeneration = true, <charging schedule>)
NotifyChargingLimitResponse()

Figure 136. Using LocalGeneration

K27 - Smart Charging with EMS and LocalGeneration - Requirements

Table 111. K27 - Requirements

ID.	Precondition	Requirements	Note
K27.FR.01	When an external charging limit/schedule is received that represents locally generated capacity	Charging Station SHALL treat this internally as a charging profile with *chargingProfilePurpose* = `LocalGeneration`	External limits can be a single value or a schedule.

K27.FR.02	K27.FR.01 AND Charging Station receives a GetChargingProfilesRequest from CSMS	Charging Station SHALL include the charging profile with *chargingProfilePurpose* = `LocalGeneration` in the reported charging profiles (if it exists).	
K27.FR.03	When the charging limit has changed by more than: `LimitChangeSignificance` and this is caused by a charging profile with *chargingProfilePurpose* = `LocalGeneration`	Charging Station SHALL send a NotifyChargingLimitRequest message to CSMS with a *chargingLimit* that has *chargingLimitSource* = `EMS` or `Other` and *isLocalGeneration* = true.	Limit can change every charging schedule period, or can be changed dynamically in a Dynamic charging profile.
K27.FR.04	K27.FR.03 AND `NotifyChargingLimitWithSchedules` is true	The NotifyChargingLimitRequest SHALL contain the charging limits/schedules as set by the external system.	
K27.FR.05	K27.FR.03 AND When Charging Station receives charging profiles from external system(s) that have a *chargingProfilePurpose* = `LocalGeneration` as well as a *chargingProfilePurpose* = `ChargingStation-ExternalConstraints`	Charging Station SHALL send the *chargingSchedules* for the charging profiles with *chargingProfilePurpose* = `LocalGeneration` together in a NotifyChargingLimitRequest message with *isLocalGeneration* = true AND Charging Station SHALL send the *chargingSchedules* for the charging profiles with *chargingProfilePurpose* = `ChargingStation-ExternalConstraints` together in a NotifyChargingLimitRequest message with *isLocalGeneration* = false or absent.	This is needed, because CSMS cannot deduce from the charging schedule that a schedule represents local generation or a limit.

5.3. ISO 15118 based Smart Charging

K15 - ISO 15118-2 Charging with load leveling

Updated in OCPP 2.1

No.	Type	Description
1	**Name**	ISO 15118-2 Charging with load leveling
2	**ID**	K15
	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	**Objective(s)**	See ISO15118-1, use case Objective E1, page 29.
4	**Description**	See ISO15118-1, use case Description E1, page 29.
	Actors	EV, Charging Station, CSMS.
	Combined scenario description	**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.
5	**Prerequisites**	Both the Charging Station and the EV support ISO 15118. The configured TxStartPoint needs to contain at least one of ParkingBayOccupied, EVConnected, Authorized or PowerPathClosed, such that the OCPP transaction is started before ChargeParameterDiscoverReq is sent by EV, such that CSMS can send a TxProfile charging profile.
6	**Postcondition(s)**	See ISO15118-1, use case End conditions E1, page 29.
7	**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.
8	**Remark(s)**	Signed SalesTariffs are supported in OCPP 2.1 via the *signatureValue* field in ChargingProfileType. Charging with ISO 15118-20 is described in use cases K18 - ISO 15118-20 Scheduled Control Mode and K19 - ISO 15118-20 Dynamic Control Mode. **NOTE**: For an ISO 15118-2 session CSMS is not recommended to send a charging profile with *chargingProfileKind* = `Dynamic` for which it intends to send frequent updates, because every update of the *limit* in the charging schedule period will trigger a renegotiation with EV for a new schedule.

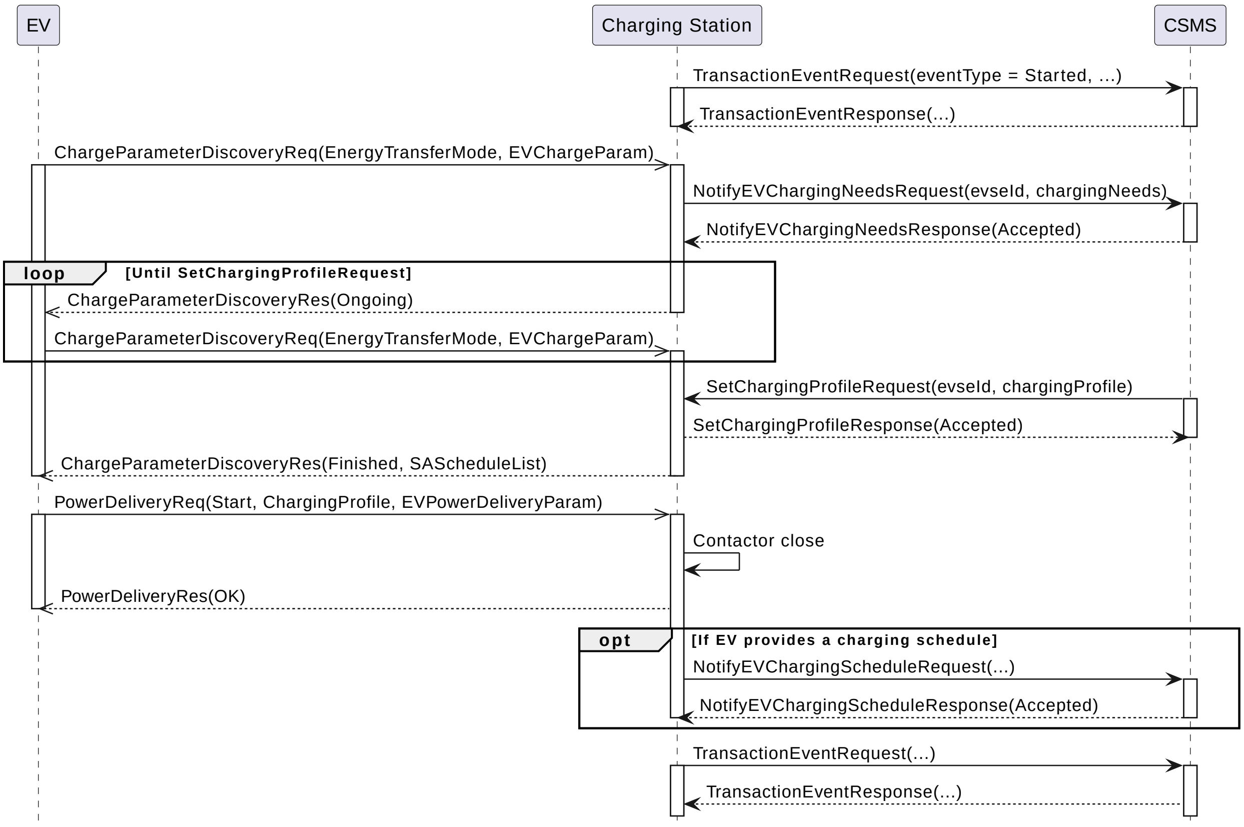

EV
Charging Station
CSMS
TransactionEventRequest(eventType = Started, ...)
TransactionEventResponse(...)
ChargeParameterDiscoveryReq(EnergyTransferMode, EVChargeParam)
NotifyEVChargingNeedsRequest(evseId, chargingNeeds)
NotifyEVChargingNeedsResponse(Accepted)
loop
[Until SetChargingProfileRequest]
ChargeParameterDiscoveryRes(Ongoing)
ChargeParameterDiscoveryReq(EnergyTransferMode, EVChargeParam)
SetChargingProfileRequest(evseId, chargingProfile)
SetChargingProfileResponse(Accepted)
ChargeParameterDiscoveryRes(Finished, SAScheduleList)
PowerDeliveryReq(Start, ChargingProfile, EVPowerDeliveryParam)
Contactor close
PowerDeliveryRes(OK)
opt
[If EV provides a charging schedule]
NotifyEVChargingScheduleRequest(...)
NotifyEVChargingScheduleResponse(Accepted)
TransactionEventRequest(...)
TransactionEventResponse(...)

Figure 137. Sequence Diagram: Charging with load leveling based on High Level Communication

K15 - ISO 15118-2 Charging with load leveling - Requirements

Table 112. 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.	See also K15.FR.20
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`.	
K15.FR.04	K15.FR.02	If the CSMS is not going to provide a charging schedule, it SHALL indicate this by setting the *status* field in the NotifyEVChargingNeedsResponse to `Rejected`.	Charging Station will use a TxDefaultProfile or provide a schedule with unlimited power.
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 *status* 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.	

ID	Precondition	Requirements	Note
K15.FR.07	K15.FR.03 or K15.FR.05	The CSMS SHALL send a SetChargingProfileRequest with *chargingProfilePurpose* = `TxProfile` and a *transactionId* and at most three *chargingSchedule* and optional *salesTariff* elements, that each contain no more periods than specified by *maxScheduleTuples* in NotifyEVChargingNeedsRequest and by device model variable `SmartChargingCtrlr.PeriodsPerSchedule`.	The Charging Station will calculate the composite schedule(s) for the EVSE (taking into account a `ChargingStationMaxProfile` or `ChargingStationExternalConstraints` 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.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.	
K15.FR.16	K15.FR.07	It is RECOMMENDED to configure the Charging Station, such that a TransactionEvent with idToken has been sent prior to the NotifyEVChargingNeedsRequest Message, so that CSMS can take the user into account when creating a charging schedule.	
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*= `InvalidMessageSeq`.	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.

ID	Precondition	Requirements	Note
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 charging schedule that was selected by the EV (i.e. the OCPP charging schedule that is represented by the SAScheduleTupleId that EV sent in the PowerDeliveryReq.)	In ISO 15118-2 the EV charging profile and the selected schedule are returned as *ChargingProfile* and *SAScheduleTupleId* in PowerDeliveryReq. See also K15.FR.21.
K15.FR.20	K15.FR.01 AND Charging Station is offline	Charging Station SHOULD add *timestamp* to the NotifyEVChargingNeedsRequest with the time when charging needs were received from EV	This will tell CSMS how old this data is, if it was not immediately sent because of an offline period.
K15.FR.21 **(2.1)**	K15.FR.10	Charging Station SHOULD set *selectedChargingScheduleId* in NotifyEVChargingScheduleRequest to the Id of the *chargingSchedule* that EV selected from the provided ChargingProfileType(s).	

K16 - Renegotiation initiated by CSMS

Updated in OCPP 2.1

No.	Type	Description
1	**Name**	Renegotiation initiated by CSMS.
2	**ID**	K16
3	**Objective(s)**	To control the charging power or current of a Charging Station
4	**Description**	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. **Note:** Description of actions between EV and Charging Station is informative only and not mandated by OCPP.
	Actors	EV, Charging Station, CSMS
	Scenario description	**1** CSMS sends a SetChargingProfileRequest to the Charging Station. **2** Charging Station responds with a SetChargingProfileResponse to the CSMS. **3** When EV sends the next CurrentDemandReq (for DC) or ChargingStatusReq (for AC), the Charging Station will respond with *evseNotification* = `ReNegotiation`. **4** EV sends a PowerDeliveryReq with *chargeProgress* = `ReNegotiate` to confirm this. **5** Charging Station responds with a PowerDeliveryRes. **6** EV sends a ChargeParameterDiscoveryReq. **7** Charging Station responds with a ChargeParameterDiscoveryRes with an SAScheduleList that contains the ChargingSchedule data from the SetChargingProfileRequest. **8** EV sends a PowerDeliveryReq with *chargeProgress* = `Start` (with an optional charging profile) to confirm this. **9** Charging Station responds with PowerDeliveryRes and, if charging was suspended at start of the renegotiation, will resume power delivery. **10** If EV provided a charging profile in the previous step, then Charging Station will send a NotifyEVChargingScheduleRequest to the CSMS.
5	**Prerequisites**	Charging session started according to use case K15.
6	**Postcondition(s)**	Charging session uses the new charging profile.
7	**Remark(s)**	Signed SalesTariffs are supported in OCPP 2.1 via the *signatureValue* field in ChargingProfileType. Charging with ISO 15118-20 is described in use cases K18 - ISO 15118-20 Scheduled Control Mode and K19 - ISO 15118-20 Dynamic Control Mode.

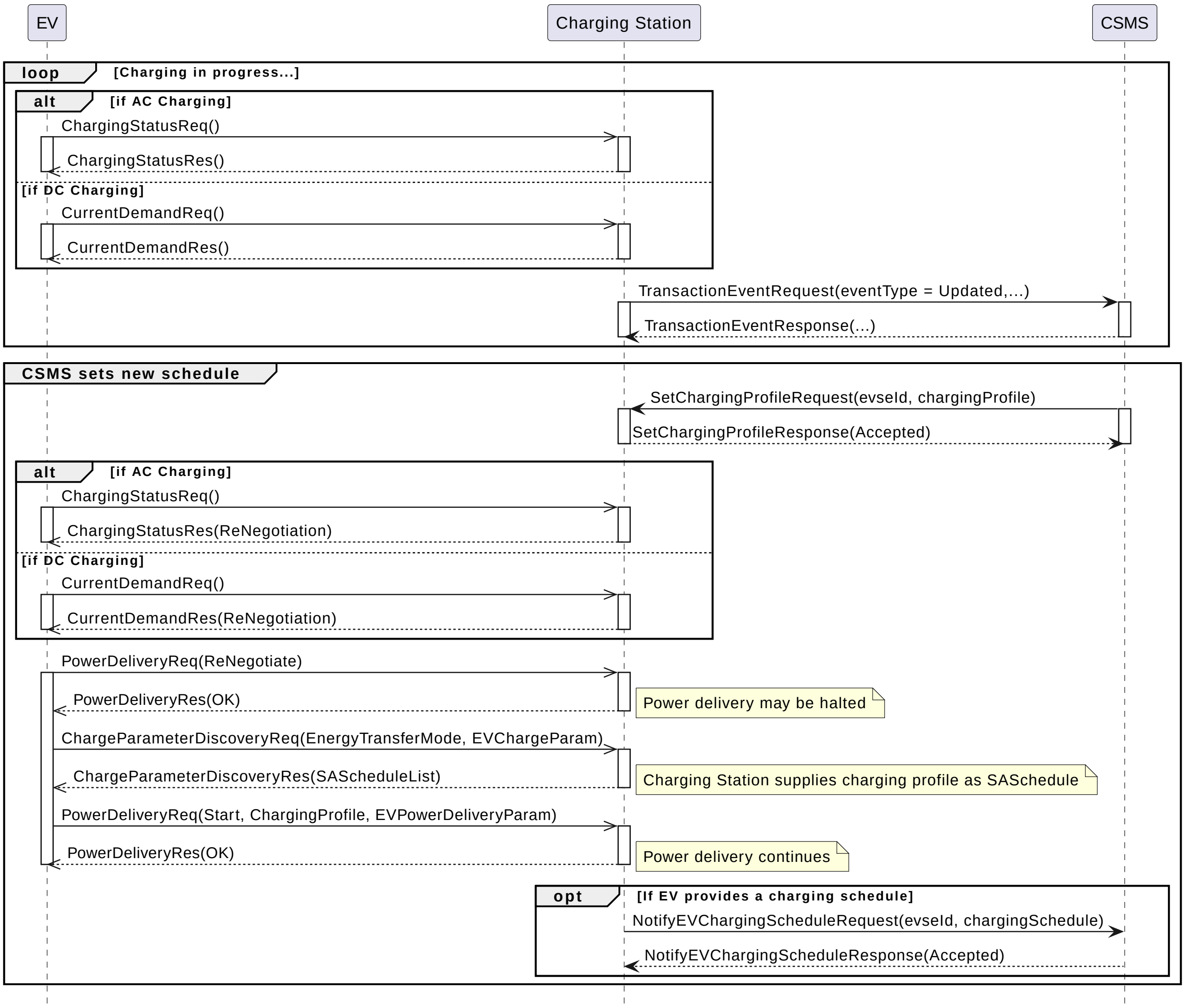

EV
Charging Station
CSMS
loop [Charging in progress...]
alt [if AC Charging]
ChargingStatusReq()
ChargingStatusRes()
[if DC Charging]
CurrentDemandReq()
CurrentDemandRes()
TransactionEventRequest(eventType = Updated,...)
TransactionEventResponse(...)
CSMS sets new schedule
SetChargingProfileRequest(evseId, chargingProfile)
SetChargingProfileResponse(Accepted)
alt [if AC Charging]
ChargingStatusReq()
ChargingStatusRes(ReNegotiation)
[if DC Charging]
CurrentDemandReq()
CurrentDemandRes(ReNegotiation)
PowerDeliveryReq(ReNegotiate)
PowerDeliveryRes(OK)
Power delivery may be halted
ChargeParameterDiscoveryReq(EnergyTransferMode, EVChargeParam)
ChargeParameterDiscoveryRes(SAScheduleList)
Charging Station supplies charging profile as SASchedule
PowerDeliveryReq(Start, ChargingProfile, EVPowerDeliveryParam)
PowerDeliveryRes(OK)
Power delivery continues
opt [If EV provides a charging schedule]
NotifyEVChargingScheduleRequest(evseId, chargingSchedule)
NotifyEVChargingScheduleResponse(Accepted)

Figure 138. Renegotiation initiated by CSMS when using ISO 15118-2

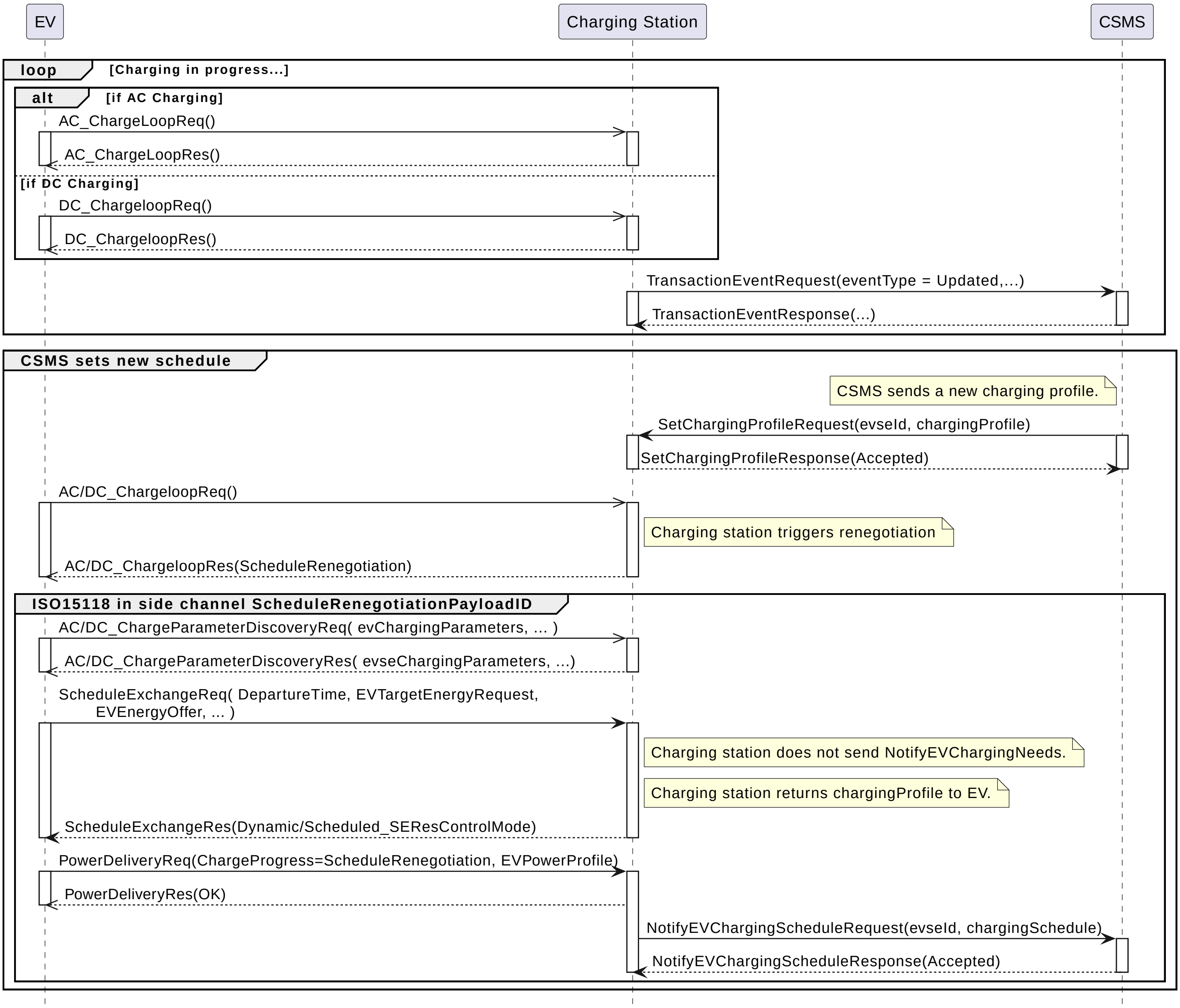

EV
Charging Station
CSMS
loop
[Charging in progress...]
alt
[if AC Charging]
AC_ChargeLoopReq()
AC_ChargeLoopRes()
[if DC Charging]
DC_ChargeloopReq()
DC_ChargeloopRes()
TransactionEventRequest(eventType = Updated,...)
TransactionEventResponse(...)
CSMS sets new schedule
CSMS sends a new charging profile.
SetChargingProfileRequest(evseId, chargingProfile)
SetChargingProfileResponse(Accepted)
AC/DC_ChargeloopReq()
Charging station triggers renegotiation
AC/DC_ChargeloopRes(ScheduleRenegotiation)
ISO15118 in side channel ScheduleRenegotiationPayloadID
AC/DC_ChargeParameterDiscoveryReq(evChargingParameters, ...)
AC/DC_ChargeParameterDiscoveryRes(evseChargingParameters, ...)
ScheduleExchangeReq(DepartureTime, EVTargetEnergyRequest, EVEnergyOffer, ...)
Charging station does not send NotifyEVChargingNeeds.
Charging station returns chargingProfile to EV.
ScheduleExchangeRes(Dynamic/Scheduled_SEResControlMode)
PowerDeliveryReq(ChargeProgress=ScheduleRenegotiation, EVPowerProfile)
PowerDeliveryRes(OK)
NotifyEVChargingScheduleRequest(evseId, chargingSchedule)
NotifyEVChargingScheduleResponse(Accepted)

Figure 139. Renegotiation initiated by CSMS when using ISO 15118-20

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 **(2.1)**	K16.FR.01	Charging Station SHALL initiate schedule renegotiation with EV.	In ISO 15118-**2** this is done by replying with EVSENotification=`ReNegotiation` to a CurrentDemandReq (for DC) or ChargingStatusReq (for AC) message. In ISO 15118-20 this is done by replying with EVSENotification=`ScheduleRenegotiation` in ChargeLoopRes.
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 **(2.1)**	EV returns a charging profile AND EV is not using dynamic control mode	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 **(2.1)**	K16.FR.04	Charging Station SHALL send the EV charging profile in a NotifyEVChargingScheduleRequest message to CSMS.	

ID	Precondition	Requirements	NOTE
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.
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 ChargeParameterDiscoveryReq.
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 **(2.1)**	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). In ISO 15118-20 this is done with EVSETargetActivePower in AC_ChargeLoopRes (for AC) and EVSEMaximumChargePower in Dynamic/Scheduled_CLResControlModeType (for DC).
K16.FR.12	K16.FR.09 AND Charging Station sends a NotifyEVChargingNeedsRequest	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 **(2.1)**	EV does not return a charging profile AND EV is not using dynamic control mode	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. the OCPP charging schedule that is represented by the SelectedScheduleTupleId that EV sent in PowerDeliveryReq).	In ISO 15118-2 the EV charging profile and the selected schedule are returned as ChargingProfile and SAScheduleTupleId in PowerDeliveryReq. In ISO 15118-20 the EV charging profile and the selected schedule are returned as EVPowerProfile and SelectedScheduleTupleId in PowerDeliveryReq. (See also K16.FR.14).
K16.FR.14 **(2.1)**	K16.FR.05	Charging Station SHOULD set *selectedChargingScheduleId* in NotifyEVChargingScheduleRequest to the Id of the *chargingSchedule* that EV selected from the provided ChargingProfileType(s).	

K17 - Renegotiation initiated by EV

Updated in OCPP 2.1

No.	Type	Description
1	**Name**	Renegotiation initiated by EV.

No.	Type	Description
2	**ID**	K16
3	**Objective(s)**	To let an EV request a new charging schedule.
4	**Description**	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. **Note:** Description of actions between EV and Charging Station is informative only and not mandated by OCPP.
	Actors	EV, Charging Station, CSMS
	Scenario description	**1** When EV sends a ChargeParameterDiscoveryReq with with charging needs parameters, then Charging Station sends this information in a NotifyEVChargingNeedsRequest to CSMS. **2** CSMS responds with NotifyEVChargingNeedsResponse to Charging Station. **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. **4** CSMS sends a SetChargingProfileRequest with the new schedule to the Charging Station. **5** Charging Station responds with SetChargingProfileResponse with status `Accepted`. **6** Charging Station sends new charging schedule to EV in a ChargeParameterDiscoveryRes message. **7** EV sends a PowerDeliveryReq with *chargeProgress* = `Start` (with an optional charging profile) to confirm this. **8** Charging Station responds with PowerDeliveryRes and, if charging was suspended at start of the renegotiation, will resume power delivery. **9** If EV provided a charging profile in the previous step, then Charging Station will send a NotifyEVChargingScheduleRequest to the CSMS.
5	**Prerequisites**	Charging session started according to use case K15.
6	**Postcondition(s)**	Charging session uses the new charging profile.
7	**Remark(s)**	Signed SalesTariffs are supported in OCPP 2.1 via the *signatureValue* field in ChargingProfileType. Charging with ISO 15118-20 is described in use cases K18 - ISO 15118-20 Scheduled Control Mode and K19 - ISO 15118-20 Dynamic Control Mode.

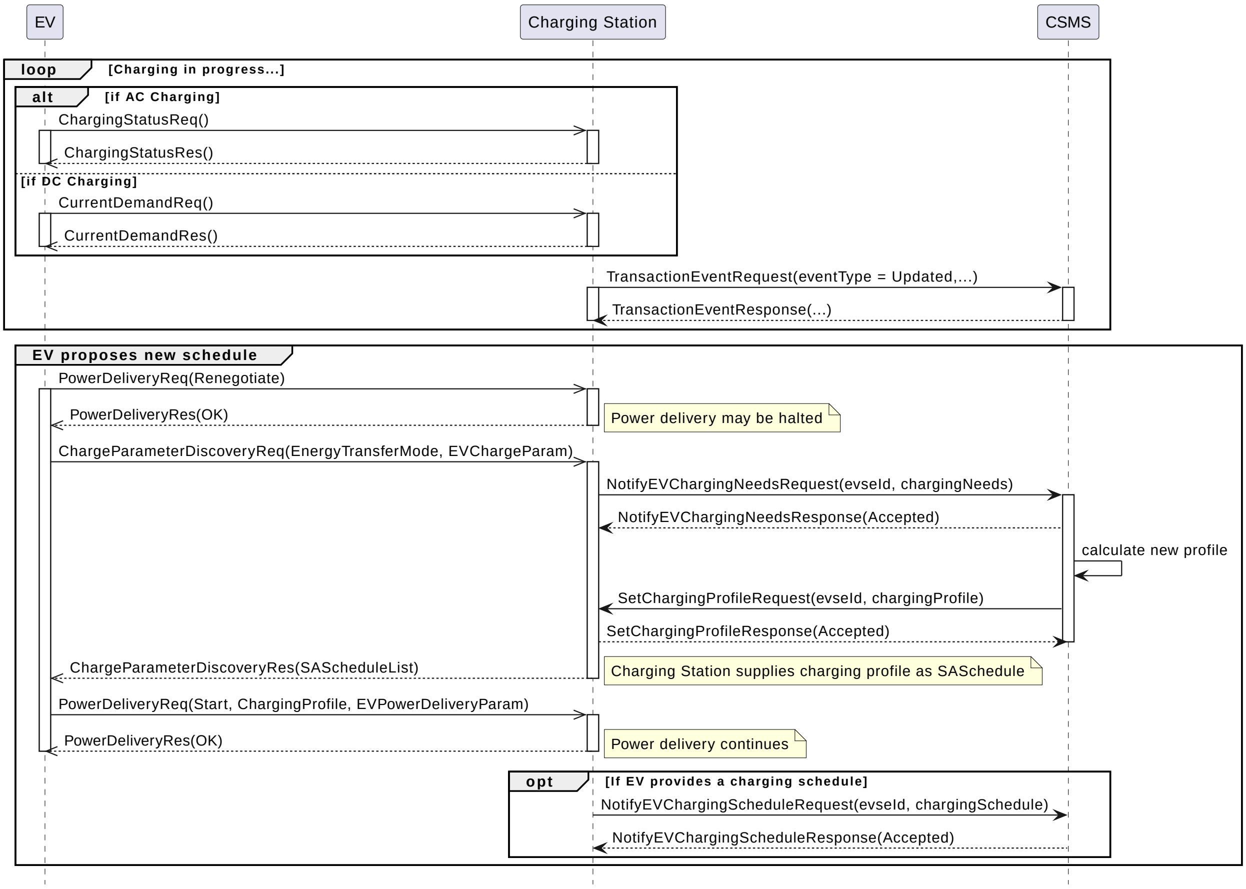

EV
Charging Station
CSMS
loop
[Charging in progress...]
alt
[if AC Charging]
ChargingStatusReq()
ChargingStatusRes()
[if DC Charging]
CurrentDemandReq()
CurrentDemandRes()
TransactionEventRequest(eventType = Updated,...)
TransactionEventResponse(...)
EV proposes new schedule
PowerDeliveryReq(Renegotiate)
PowerDeliveryRes(OK)
Power delivery may be halted
ChargeParameterDiscoveryReq(EnergyTransferMode, EVChargeParam)
NotifyEVChargingNeedsRequest(evseId, chargingNeeds)
NotifyEVChargingNeedsResponse(Accepted)
calculate new profile
SetChargingProfileRequest(evseId, chargingProfile)
SetChargingProfileResponse(Accepted)
ChargeParameterDiscoveryRes(SAScheduleList)
Charging Station supplies charging profile as SASchedule
PowerDeliveryReq(Start, ChargingProfile, EVPowerDeliveryParam)
PowerDeliveryRes(OK)
Power delivery continues
opt
[If EV provides a charging schedule]
NotifyEVChargingScheduleRequest(evseId, chargingSchedule)
NotifyEVChargingScheduleResponse(Accepted)

Figure 140. Renegotiation initiated by EV when using ISO 15118-2

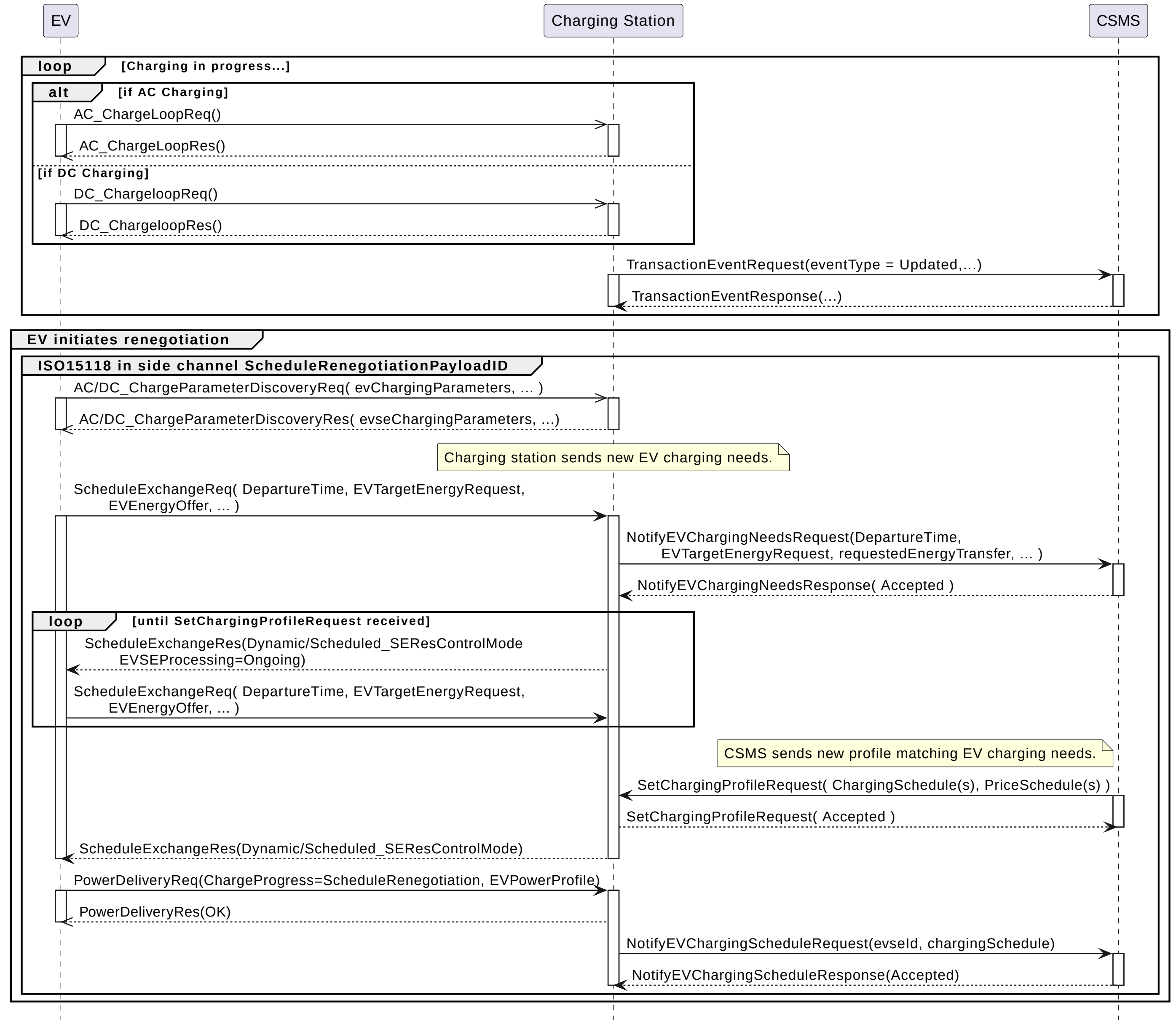

EV
Charging Station
CSMS
loop [Charging in progress...]
alt [if AC Charging]
AC_ChargeLoopReq()
AC_ChargeLoopRes()
[if DC Charging]
DC_ChargeloopReq()
DC_ChargeloopRes()
TransactionEventRequest(eventType = Updated,...)
TransactionEventResponse(...)
EV initiates renegotiation
ISO15118 in side channel ScheduleRenegotiationPayloadID
AC/DC_ChargeParameterDiscoveryReq(evChargingParameters, ...)
AC/DC_ChargeParameterDiscoveryRes(evseChargingParameters, ...)
Charging station sends new EV charging needs.
ScheduleExchangeReq(DepartureTime, EVTargetEnergyRequest, EVEnergyOffer, ...)
NotifyEVChargingNeedsRequest(DepartureTime, EVTargetEnergyRequest, requestedEnergyTransfer, ...)
NotifyEVChargingNeedsResponse(Accepted)
loop [until SetChargingProfileRequest received]
ScheduleExchangeRes(Dynamic/Scheduled_SEResControlMode EVSEProcessing=Ongoing)
ScheduleExchangeReq(DepartureTime, EVTargetEnergyRequest, EVEnergyOffer, ...)
CSMS sends new profile matching EV charging needs.
SetChargingProfileRequest(ChargingSchedule(s), PriceSchedule(s))
SetChargingProfileRequest(Accepted)
ScheduleExchangeRes(Dynamic/Scheduled_SEResControlMode)
PowerDeliveryReq(ChargeProgress=ScheduleRenegotiation, EVPowerProfile)
PowerDeliveryRes(OK)
NotifyEVChargingScheduleRequest(evseId, chargingSchedule)
NotifyEVChargingScheduleResponse(Accepted)

Figure 141. Renegotiation initiated by EV when using ISO 15118-20

K17 - Renegotiation initiated by EV - Requirements

Table 113. 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.	
K17.FR.02	K17.FR.01	In response to a NotifyEVChargingNeedsRequest the CSMS SHALL send a NotifyEVChargingNeedsResponse.	
K17.FR.03	K17.FR.02 AND If the CSMS is able to provide a charging schedule now	CSMS SHALL indicate this by setting the *status* field in the NotifyEVChargingNeedsResponse to 'Accepted'.	
K17.FR.04	K17.FR.02 AND If the CSMS is not able to provide a charging schedule now	CSMS SHALL indicate this by setting the *status* field in the NotifyEVChargingNeedsResponse to 'Rejected'.	

ID	Precondition	Requirements	Note
K17.FR.05	K17.FR.02 AND If the CSMS is able to provide a charging schedule, but needs processing time	CSMS SHALL indicate this by setting the *status* field in the NotifyEVChargingNeedsResponse to 'Processing'.	
K17.FR.06 **(2.1)**		A NotifyEVChargingNeedsRequest SHALL contain either *ACChargingParameters* or *DCChargingParameters* when Charging Station is using ISO 15118-2, or *V2XChargingParameters* when using ISO 15118-20.	
K17.FR.07 **(2.1)**	(K17.FR.03 OR K17.FR.05) AND *controlMode* is `ScheduledControl` or absent	The CSMS SHALL send a SetChargingProfileRequest with *chargingProfilePurpose* = `TxProfile` and at most three *chargingSchedule* and optional price schedule elements, that each contain no more periods than specified by *maxScheduleTuples* 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 **(2.1)**	K17.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. Not applicable in dynamic control mode.
K17.FR.12 **(2.1)**	K17.FR.10 AND EV charging profile is NOT within limits of CSMS ChargingSchedule	CSMS responds with NotifyEVChargingScheduleResponse with *status* `Rejected` to Charging Station.	Not applicable in dynamic control mode.
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.	
K17.FR.15	K17.FR.01 AND (K17.FR.04 OR 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.	

ID	Precondition	Requirements	Note
K17.FR.16 **(2.1)**	K17.FR.07 EV does not return a charging profile AND EV is not using dynamic control mode	Charging Station IS RECOMMENDED to return an EV charging profile as a chargingSchedule in a NotifyEVChargingScheduleRequest message to CSMS that matches the charging schedule that was selected by the EV (i.e. the OCPP charging schedule that is represented by the SelectedScheduleTupleId that EV sent in PowerDeliveryReq).	In ISO 15118-2 the EV charging profile and the selected schedule are returned as ChargingProfile and SAScheduleTupleId in PowerDeliveryReq. In ISO 15118-20 the EV charging profile and the selected schedule are returned as EVPowerProfile and SelectedScheduleTupleId. (See also K17.FR.18)
K17.FR.17	K17.FR.01 AND Charging Station is offline	Charging Station SHOULD add *timestamp* to the NotifyEVChargingNeedsRequest with the time when charging needs were received from EV	This will tell CSMS how old this data is, if it was not immediately sent because of an offline period.
K17.FR.18 **(2.1)**	K17.FR.10	Charging Station SHOULD set *selectedChargingScheduleId* in NotifyEVChargingScheduleRequest to the Id of the *chargingSchedule* that EV selected from the provided ChargingProfileType(s).	
K17.FR.19 **(2.1)**	(K17.FR.03 OR K17.FR.05) AND *controlMode* is `DynamicControl`	The CSMS SHALL send a SetChargingProfileRequest with *chargingProfilePurpose* = `TxProfile` and a *transactionId* and one *chargingSchedule* element with one or more *chargingSchedulePeriods*, and optionally an *absolutePriceSchedule* or *priceLevelSchedule* element, that each contain no more periods than specified by *maxScheduleTuples* in NotifyEVChargingNeedsRequest and by device model variable `SmartChargingCtrlr.PeriodsPerSchedule`.	The Charging Station will calculate the composite schedule for the EVSE (taking into account a `ChargingStationMaxProfile` or `ChargingStationExternalConstraints` if present).

K18 - ISO 15118-20 Scheduled Control Mode

New in OCPP 2.1

No.	Type	Description
1	**Name**	ISO 15118-20 Scheduled Control Mode
2	**ID**	K18
3	**Objective(s)**	To describe the charging process when Charging Stations uses ISO 15118-20 with scheduled control mode
4	**Description**	CSMS provides a charging profile with up to three charging schedules and optional price information that try to match the energy needs of EV. EV selects a charging schedule to follow and returns its calculated charging schedule back to Charging Station.
	Actors	EV, Charging Station, CSMS.

No.	Type	Description
	Scenario description	1. EV and Charging Station exchange charge parameters with an AC/DC_ChargeParameterDiscovery message. 2. EV sends energy needs and departure time to Charging Station with a ScheduleExchangeReq message. 3. Charging Station sends a NotifyEVChargingNeedsRequest message to CSMS with *controlMode* = `ScheduledControl`. 4. CSMS responds with *status* = `Accepted`. 5. CSMS sends a charging profile with one or more charging schedules and optional price information to Charging Station via a SetChargingProfileRequest. 6. Charging station responds with *status* = `Accepted`. 7. Charging Station returns offered charging schedule(s) to EV via the ScheduleExchangeRes response. 8. EV selects a charging schedule to follow, and calculates its expected charging profile in EVPowerProfile. 9. EV sends a PowerDeliveryReq with Id of the selected schedule to follow and its EVPowerProfile to Charging Station. 10. Charging Station sends this Id an EVPowerProfile as a NotifyEVChargingScheduleRequest to CSMS.
5	**Prerequisites**	Both the Charging Station and the EV support ISO 15118-20 Scheduled Control Mode.
6	**Postcondition(s)**	
7	**Error handling**	Charging Station needs to use the information from the SetChargingProfileRequest message from CSMS to create the response to the ISO 15118-20 ScheduleExchangeReq message from 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 ScheduleExchangeRes that matches charging profiles that are already active at EVSE, e.g. TxDefaultProfile or ChargingStationMaxProfile. When CSMS sends the SetChargingProfileRequest at a later time, then this will trigger a renegotiation according to use case K16 - Renegotiation initiated by CSMS.
8	**Remark(s)**	Signed PriceSchedules (ISO 15118-20) are supported in OCPP 2.1 via the *signatureValue* field in ChargingProfileType. When the Charging Station uses ISO 15118-20 "scheduled control mode", it sends charging schedules to the EV. The EV selects one and returns its calculated EVPowerProfile that fits in selected schedule to Charging Station in a PowerDeliveryReq message. Charging Station sends this to CSMS in a NotifyEVChargingScheduleRequest. **NOTE**: When NotifyEVChargingNeedsRequest has *controlMode* = `ScheduledControl` CSMS is not recommended to send a charging profile with *chargingProfileKind* = `Dynamic` for which it intends to send frequent updates, because every update of the *limit* in the charging schedule period will trigger a renegotiation with EV for a new schedule.

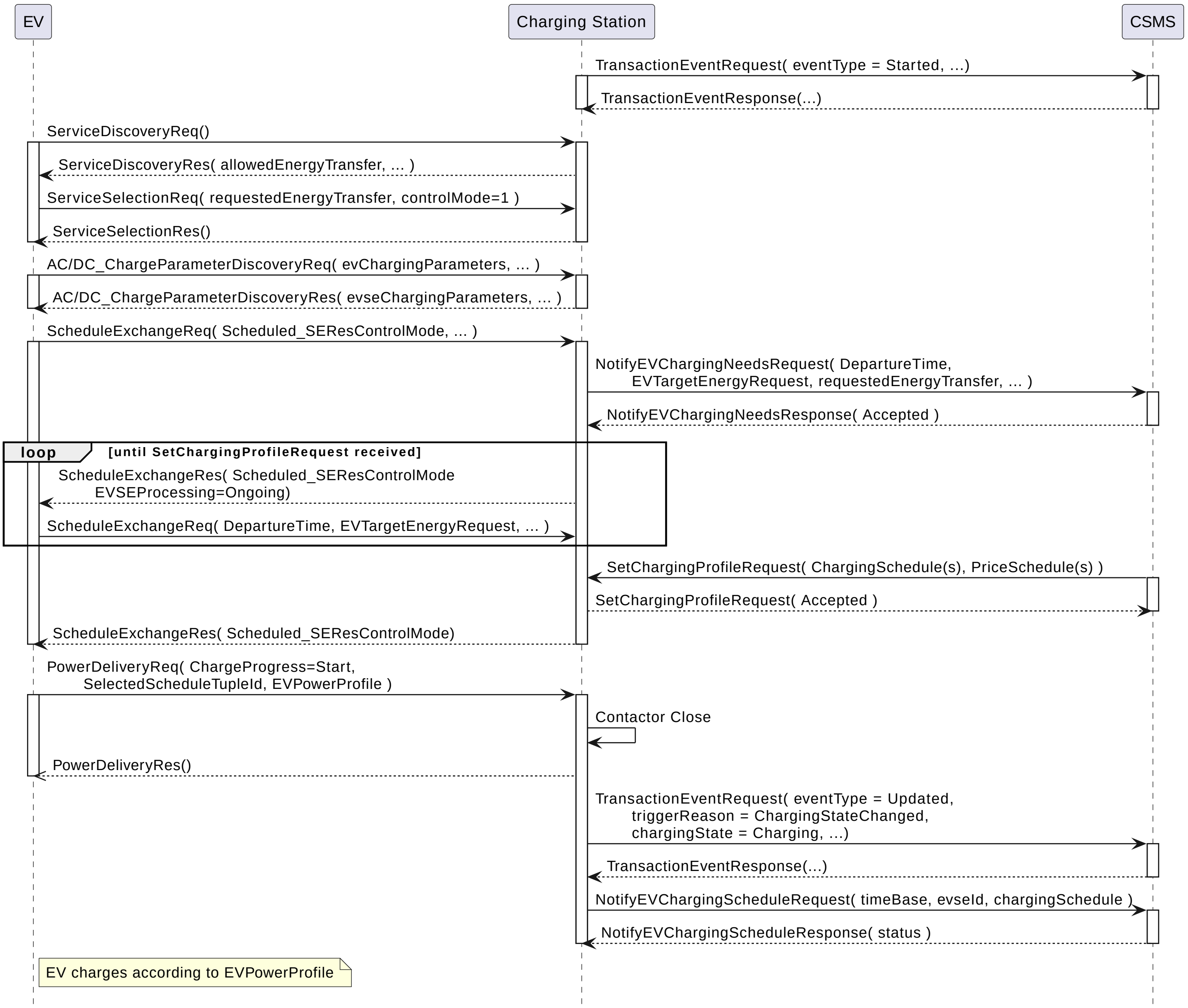

EV
Charging Station
CSMS
TransactionEventRequest(eventType = Started, ...)
TransactionEventResponse(...)
ServiceDiscoveryReq()
ServiceDiscoveryRes(allowedEnergyTransfer, ...)
ServiceSelectionReq(requestedEnergyTransfer, controlMode=1)
ServiceSelectionRes()
AC/DC_ChargeParameterDiscoveryReq(evChargingParameters, ...)
AC/DC_ChargeParameterDiscoveryRes(evseChargingParameters, ...)
ScheduleExchangeReq(Scheduled_SEResControlMode, ...)
NotifyEVChargingNeedsRequest(DepartureTime, EVTargetEnergyRequest, requestedEnergyTransfer, ...)
NotifyEVChargingNeedsResponse(Accepted)
loop
[until SetChargingProfileRequest received]
ScheduleExchangeRes(Scheduled_SEResControlMode EVSEProcessing=Ongoing)
ScheduleExchangeReq(DepartureTime, EVTargetEnergyRequest, ...)
SetChargingProfileRequest(ChargingSchedule(s), PriceSchedule(s))
SetChargingProfileRequest(Accepted)
ScheduleExchangeRes(Scheduled_SEResControlMode)
PowerDeliveryReq(ChargeProgress=Start, SelectedScheduleTupleId, EVPowerProfile)
Contactor Close
PowerDeliveryRes()
TransactionEventRequest(eventType = Updated, triggerReason = ChargingStateChanged, chargingState = Charging, ...)
TransactionEventResponse(...)
NotifyEVChargingScheduleRequest(timeBase, evseId, chargingSchedule)
NotifyEVChargingScheduleResponse(status)
EV charges according to EVPowerProfile

Figure 142. Charging with load-leveling using ISO 15118-20 Scheduled Control Mode

K18 - ISO 15118-20 Scheduled Control Mode - Requirements

Table 114. K18 - Requirements

ID	Precondition	Requirements	Note
K18.FR.01	When the Charging Station receives charging needs from the EV in ScheduleExchangeReq for scheduled control mode	The Charging Station SHALL send a NotifyEVChargingNeedsRequest with *controlMode* = `ScheduledControl` to the CSMS.	
K18.FR.02	K18.FR.01	In response to a NotifyEVChargingNeedsRequest the CSMS SHALL send a NotifyEVChargingNeedsResponse.	
K18.FR.03	K18.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`.	
K18.FR.04	K18.FR.02	If the CSMS is not able to provide a charging schedule, it SHALL indicate this by setting the *status* field in the NotifyEVChargingNeedsResponse to `NoChargingProfile`.	(Note, *status* value differs from K15.FR.04). Charging Station will use a TxDefaultProfile or provide a schedule with unlimited power.

ID	Precondition	Requirements	Note
K18.FR.05	K18.FR.02	If the CSMS is able to provide a charging schedule; but needs processing time, it SHALL indicate this by setting the *status* 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.
K18.FR.06		The NotifyEVChargingNeedsRequest SHALL contain V2XChargingParametersType instead of ACChargingParametersType or DCChargingParametersType	ISO 15118-20 charging parameters are contained in V2XChargingParametersType.
K18.FR.07	K18.FR.01 AND (K18.FR.03 OR K18.FR.05)	The CSMS SHALL send a SetChargingProfileRequest with *chargingProfilePurpose* = `TxProfile` and a *transactionId* and at most three *chargingSchedule* elements with one or more *chargingSchedulePeriods* (at most `SmartChargingCtrlr.PeriodsPerSchedule` periods), and optionally an *absolutePriceSchedule* or *priceLevelSchedule* element, that each contain no more entries than specified by *maxScheduleTuples* in NotifyEVChargingNeedsRequest.	The Charging Station will calculate the composite schedule(s) for the EVSE (taking into account a `ChargingStationMaxProfile` or `ChargingStationExternalConstraints` if present) and will convert that to the ScheduleTupleType format for ISO 15118-20. Maximum number of entries is determined by MaximumSupportedPoints from ScheduleExchangeReq message.
K18.FR.08	K18.FR.01	The CSMS SHOULD send a SetChargingProfileRequest to the Charging Station within 60 seconds.	This is to satisfy the ISO 15118-20 ScheduleExchangeReq timeout. See also K18.FR.20
K18.FR.09	K18.FR.20 AND EV returns the Id of the selected charging schedule and an EVPowerProfile in PowerDeliveryReq	Charging Station SHALL verify that provided charging profile is within boundaries of the selected ChargingSchedule from CSMS.	
K18.FR.10	K18.FR.09	Charging Station SHALL send the EVPowerProfile from EV as a *chargingSchedule* in a NotifyEVChargingScheduleRequest message to CSMS and set the field *selectedChargingScheduleId* to the Id of the charging schedule selected by EV.	
K18.FR.11	K18.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.
K18.FR.12	K18.FR.10 AND *chargingSchedule* in NotifyEVChargingScheduleRequest is NOT within limits of CSMS ChargingSchedule	CSMS responds with NotifyEVChargingScheduleResponse with *status* `Rejected` to Charging Station.	
K18.FR.13	K18.FR.12	CSMS starts new renegotiation as per use case K16.	
K18.FR.14	K18.FR.11	The Charging Station SHOULD take the schedule from the NotifyEVChargingScheduleRequest into account when calculating the actual composite schedule.	
K18.FR.15	K18.FR.03 AND (K18.FR.04 OR 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.	

ID	Precondition	Requirements	Note
K18.FR.16	K18.FR.07	It is RECOMMENDED to configure the Charging Station, such that a TransactionEvent with idToken has been sent prior to the NotifyEVChargingNeedsRequest Message, so that CSMS can take the user into account when creating a charging schedule.	This can be achieved by setting TxStartPoint to Authorized, EVConnected or PowerPathClosed.
K18.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.
K18.FR.18	K18.FR.03 OR K18.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.
K18.FR.19	K18.FR.07 AND EV does not return EVPowerProfile	Charging Station IS RECOMMENDED to return an EV charging profile as a *chargingSchedule* in a NotifyEVChargingScheduleRequest message to CSMS that matches the charging schedule that was selected by the EV (i.e. the OCPP charging schedule that is represented by the SelectedScheduleTupleId that EV sent in the PowerDeliveryReq.)	This situation should not occur, because EV is required to send an EVPowerProfile.
K18.FR.20	K18.FR.08	The Charging Station SHALL return the charging (and optional price) schedule(s) to EV in the ScheduleExchangeRes response.	
K18.FR.21	K18.FR.01 AND Charging Station is offline	Charging Station SHOULD add *timestamp* to the NotifyEVChargingNeedsRequest with the time when charging needs were received from EV	This will tell CSMS how old this data is, if it was not immediately sent because of an offline period.
K18.FR.22	K18.FR.02 AND CSMS does not accept the *requestedEnergyTransfer* in NotifyEVChargingNeedsRequest	CSMS SHALL respond with NotifyEVChargingNeedsResponse with *status* = `Rejected`.	Charging station will then stop the transaction.
K18.FR.23	K18.FR.22	Charging Station SHALL terminate the transaction and send a TransactionEventRequest with *eventType* = `Ended`, *triggerReason* = `AbnormalCondition`, *stoppedReason* = `ReqEnergyTransferRejected`.	

K19 - ISO 15118-20 Dynamic Control Mode

New in OCPP 2.1

No.	Type	Description
1	**Name**	ISO 15118-20 Dynamic Control Mode
2	**ID**	K19
3	**Objective(s)**	To describe the charging process when Charging Stations uses ISO 15118-20 with dynamic control mode
4	**Description**	CSMS provides a charging profile with one charging schedule and optional price information that tries to match the energy needs of EV. Charging Station only sends the pricing schedule to EV. EV returns its fastest possible charging schedule back to Charging Station.
	Actors	EV, Charging Station, CSMS.

No.	Type	Description
	Scenario description	1. EV and Charging Station exchange charge parameters with an AC/DC_ChargeParameterDiscovery message. 2. EV sends energy needs and departure time to Charging Station with a ScheduleExchangeReq message. 3. Charging Station sends a NotifyEVChargingNeedsRequest message with *controlMode* = `DynamicControl` to CSMS. 4. CSMS responds with *status* = `Accepted`. 5. CSMS sends a charging profile with one charging schedule and optional price information to Charging Station via a SetChargingProfileRequest. 6. Charging station responds with *status* = `Accepted`. 7. Charging Station returns the offered pricing schedule (but not the charging schedule) to EV via a ScheduleExchangeRes response. 8. EV sends a PowerDeliveryReq with its EVPowerProfile that describes its fastest possible charging schedule to Charging Station. 9. Charging Station ignores this EVPowerProfile and sends the charging schedule provided by CSMS as a NotifyEVChargingScheduleRequest to CSMS.
5	**Prerequisites**	Both the Charging Station and the EV support ISO 15118-20 Dynamic Control Mode.
6	**Postcondition(s)**	
7	**Error handling**	Charging Station needs to use the information from the SetChargingProfileRequest message from CSMS to create the response to the ISO 15118-20 ScheduleExchangeReq message from 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 ScheduleExchangeRes that matches charging profiles that are already active at EVSE, e.g. TxDefaultProfile or ChargingStationMaxProfile. When CSMS sends the SetChargingProfileRequest at a later time, then this will trigger a renegotiation according to use case K16 - Renegotiation initiated by CSMS.
8	**Remark(s)**	Signed PriceSchedules (ISO 15118-20) are supported in OCPP 2.1 via the *signatureValue* field in ChargingProfileType. When the Charging Station uses ISO 15118-20 "dynamic control mode", it does not send any charging schedules to the EV (only an optional price schedule). Instead, it will dynamically update the power limit over time to match the charging profile from CSMS. The EV still returns an EVPowerProfile to the Charging Station in PowerDeliveryReq. This EVPowerProfile is the fastest way the EV can charge given the charge parameters, and it disregards the charging profile provided to the Charging Station by CSMS. Therefore this should not be returned as a NotifyEVChargingSchedule to CSMS. Instead, Charging Station returns the CSMS charging schedule in NotifyEVChargingSchedule, since this is the schedule that EV will be made to follow. **NOTE**: ISO 15118-20 "dynamic control mode" does not require a charging profile from CSMS with *chargingProfileKind* = `Dynamic`. For example, CSMS can send an `Absolute` charging profile with several charging schedule periods, that the Charging Station executes towards the EV using dynamic control mode.

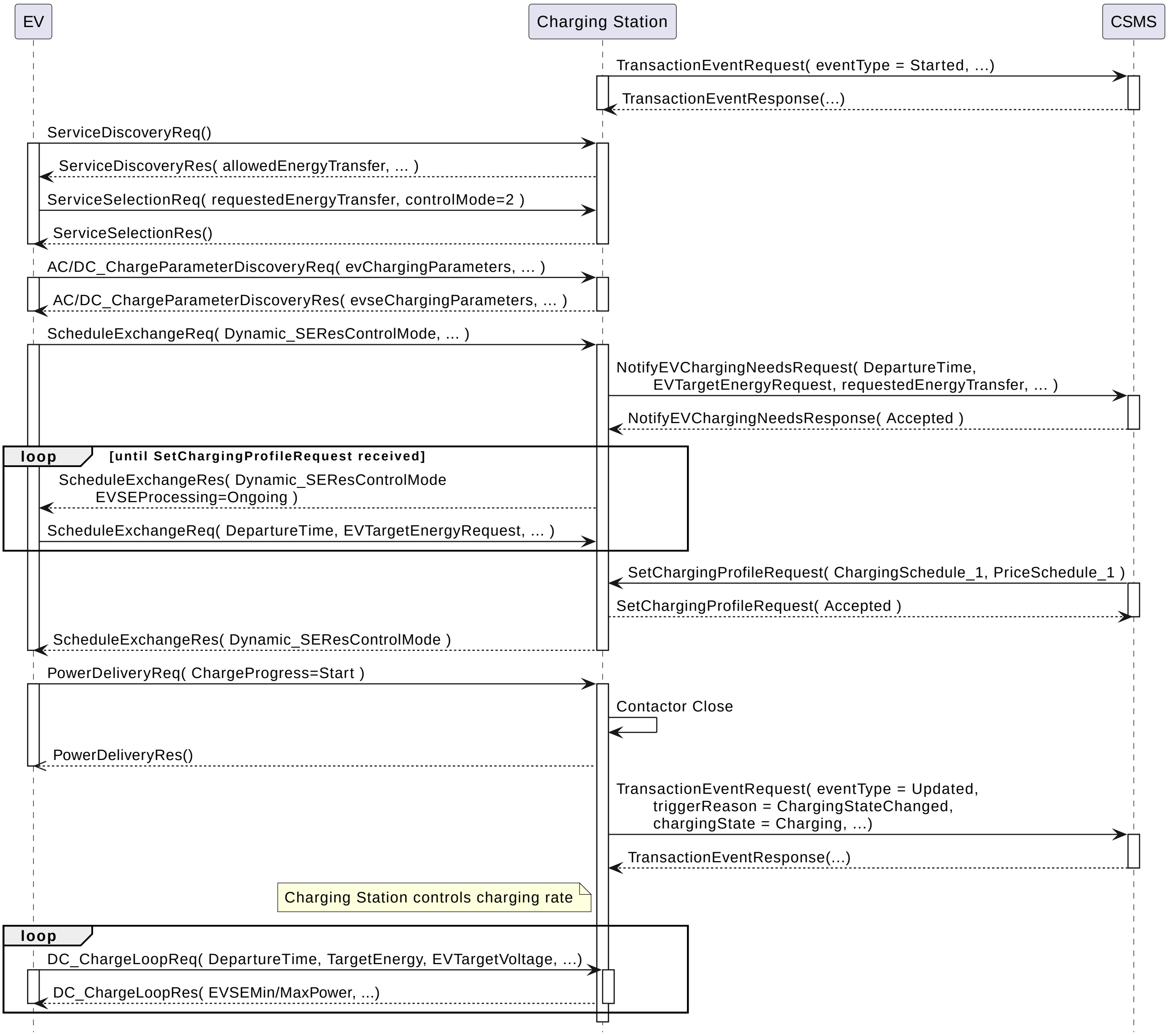

EV
Charging Station
CSMS
TransactionEventRequest(eventType = Started, ...)
TransactionEventResponse(...)
ServiceDiscoveryReq()
ServiceDiscoveryRes(allowedEnergyTransfer, ...)
ServiceSelectionReq(requestedEnergyTransfer, controlMode=2)
ServiceSelectionRes()
AC/DC_ChargeParameterDiscoveryReq(evChargingParameters, ...)
AC/DC_ChargeParameterDiscoveryRes(evseChargingParameters, ...)
ScheduleExchangeReq(Dynamic_SEResControlMode, ...)
NotifyEVChargingNeedsRequest(DepartureTime, EVTargetEnergyRequest, requestedEnergyTransfer, ...)
NotifyEVChargingNeedsResponse(Accepted)
loop
[until SetChargingProfileRequest received]
ScheduleExchangeRes(Dynamic_SEResControlMode EVSEProcessing=Ongoing)
ScheduleExchangeReq(DepartureTime, EVTargetEnergyRequest, ...)
SetChargingProfileRequest(ChargingSchedule_1, PriceSchedule_1)
SetChargingProfileRequest(Accepted)
ScheduleExchangeRes(Dynamic_SEResControlMode)
PowerDeliveryReq(ChargeProgress=Start)
Contactor Close
PowerDeliveryRes()
TransactionEventRequest(eventType = Updated, triggerReason = ChargingStateChanged, chargingState = Charging, ...)
TransactionEventResponse(...)
Charging Station controls charging rate
loop
DC_ChargeLoopReq(DepartureTime, TargetEnergy, EVTargetVoltage, ...)
DC_ChargeLoopRes(EVSEMin/MaxPower, ...)

Figure 143. Charging with load-leveling using ISO 15118-20 Dynamic Control Mode

K19 - ISO 15118-20 Dynamic Control Mode - Requirements

Table 115. K19 - Requirements

ID	Precondition	Requirements	Note
K19.FR.01	When the Charging Station receives charging needs from the EV in ScheduleExchangeReq for dynamic control mode	The Charging Station SHALL send a NotifyEVChargingNeedsRequest with *controlMode* = `DynamicControl` to the CSMS.	
K19.FR.02	K19.FR.01	In response to a NotifyEVChargingNeedsRequest the CSMS SHALL send a NotifyEVChargingNeedsResponse.	
K19.FR.03	K19.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`.	
K19.FR.04	K19.FR.02	If the CSMS is not able to provide a charging schedule, it SHALL indicate this by setting the *status* field in the NotifyEVChargingNeedsResponse to `NoChargingProfile`.	(Note, *status* value differs from K15.FR.04). Charging Station will use a TxDefaultProfile or provide a schedule with unlimited power.

ID	Precondition	Requirements	Note
K19.FR.05	K19.FR.02	If the CSMS is able to provide a charging schedule; but needs processing time, it SHALL indicate this by setting the *status* 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.
K19.FR.06		The NotifyEVChargingNeedsRequest SHALL contain V2XChargingParametersType instead of ACChargingParametersType or DCChargingParametersType	ISO 15118-20 charging parameters are contained in V2XChargingParametersType.
K19.FR.07	K19.FR.01 AND (K19.FR.03 OR K19.FR.05)	The CSMS SHALL send a SetChargingProfileRequest with *chargingProfilePurpose* = `TxProfile` and a *transactionId* and one *chargingSchedule* element with one or more *chargingSchedulePeriods*, and optionally an *absolutePriceSchedule* or *priceLevelSchedule* element, that in total contain no more periods than specified by *maxScheduleTuples* in NotifyEVChargingNeedsRequest and by device model variable `SmartChargingCtrlr.PeriodsPerSchedule`.	The Charging Station will calculate the composite schedule for the EVSE (taking into account a `ChargingStationMaxProfile` or `ChargingStationExternalConstraints` if present). *maxScheduleTuples* provides a maximum for charging schedule periods, price schedule entries and price rules. (see ISO 15118-20 [V2G20-2175]).
K19.FR.08	K19.FR.01	The CSMS SHOULD send a SetChargingProfileRequest to the Charging Station within 60 seconds.	This is to satisfy the ISO 15118-20 ScheduleExchangeReq timeout.
K19.FR.09	K19.FR.08	The Charging Station SHALL return the optional price schedule to EV in the ScheduleExchangeRes response.	
K19.FR.10	EV sends an EVPowerProfile in PowerDeliveryReq	Charging Station SHALL disregard the EVPowerProfile from EV and use the charging schedule received from CSMS as the *chargingSchedule* in the NotifyEVChargingScheduleRequest message to CSMS and set the field *selectedChargingScheduleId* to the Id of this charging schedule.	Charging Station will ensure that EV follows the CSMS charging schedule.
K19.FR.11	K19.FR.01 AND (K19.FR.04 OR 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.	
K19.FR.12	K19.FR.07	It is RECOMMENDED to configure the Charging Station, such that a TransactionEvent with idToken has been sent prior to the NotifyEVChargingNeedsRequest Message, so that CSMS can take the user into account when creating a charging schedule.	This can be achieved by setting TxStartPoint to Authorized, EVConnected or PowerPathClosed.
K19.FR.13	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.
K19.FR.14	K19.FR.01 AND Charging Station is offline	Charging Station SHOULD add *timestamp* to the NotifyEVChargingNeedsRequest with the time when charging needs were received from EV	This will tell CSMS how old this data is, if it was not immediately sent because of an offline period.
K19.FR.15	K19.FR.02 AND CSMS does not accept the *requestedEnergyTransfer* in NotifyEVChargingNeedsRequest	CSMS SHALL respond with NotifyEVChargingNeedsResponse with *status* = `Rejected`.	Charging station will then stop the transaction.

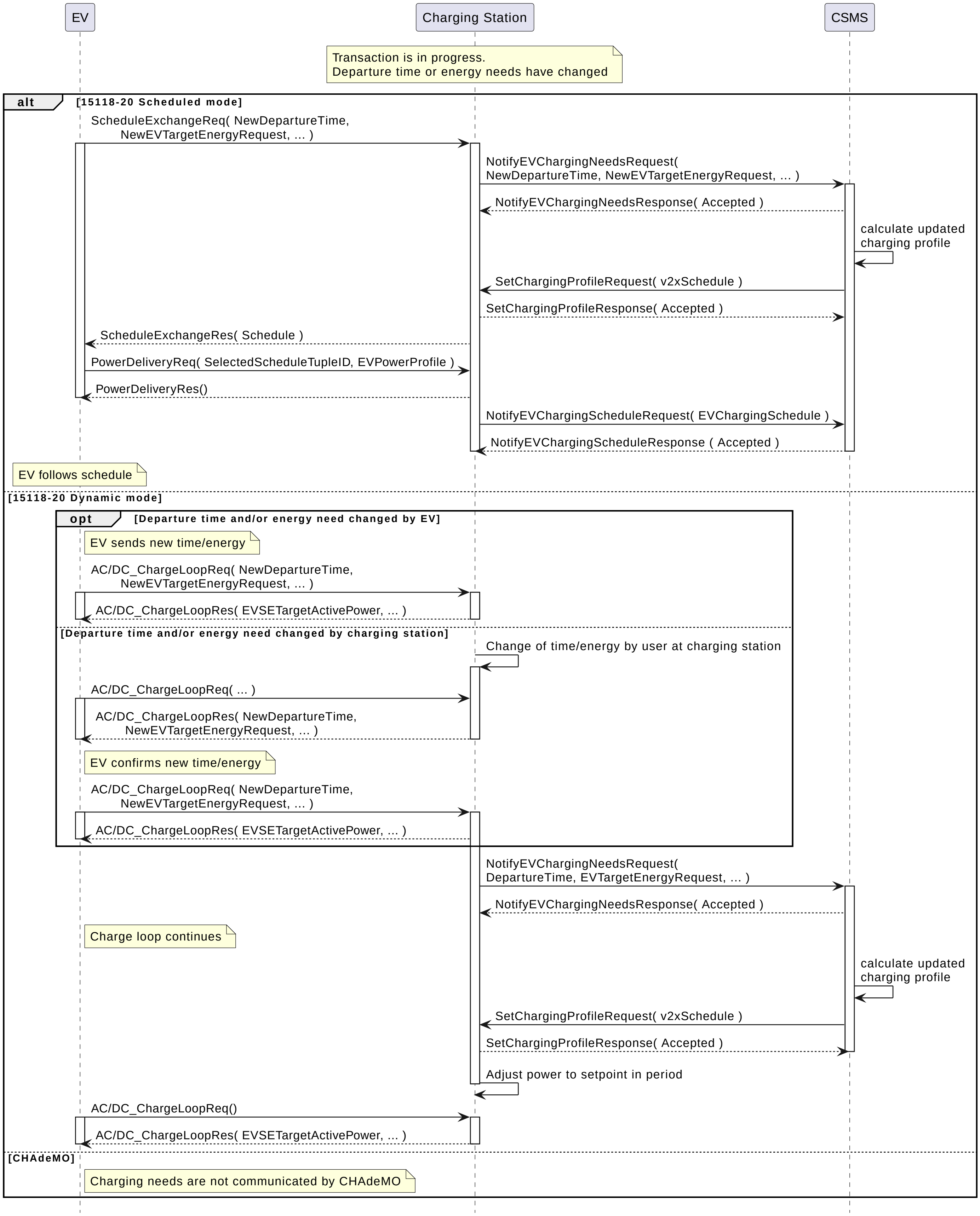
ID	Precondition	Requirements	Note
K19.FR.16	K19.FR.15	Charging Station SHALL terminate the transaction and send a TransactionEventRequest with *eventType* = `Ended`, *triggerReason* = `AbnormalCondition`, *stoppedReason* = `ReqEnergyTransferRejected`.	

K20 - ISO 15118-20 Adjusting charging schedule when energy needs change

New in OCPP 2.1

No.	Type	Description
1	**Name**	ISO 15118-20 Adjusting charging schedule when energy needs change
2	**ID**	K20
3	**Objective(s)**	To adjust the charging schedule when EV reports a change in departure time or energy need.
4	**Description**	The user has changed the departure time or requested energy amount, either via EV or charging station. This means that the charging profile may have to be updated for these needs. The charging station notifies CSMS of this situation.
	Actors	EV, Charging Station, CSMS
	Scenario description #1	Via EV when using ISO 15118-20 Scheduled Control mode 1. EV sends a ScheduleExchangeReq message with new DepartureTime and/or EVTargetEnergyRequest. 2. CS sends a NotifyEVChargingNeedsRequest with given *departureTime* and *evTargetEnergyRequest* to CSMS. 3. CSMS calculates a new charging profile and sends a SetChargingProfileRequest to the Charging Station. 4. CS returns the new charging schedule to EV in the ScheduleExchangeRes response message. 5. EV sends its projected charging schedule (EVPowerProfile) in a PowerDeliveryReq to the Charging Station. 6. CS sends this EV charging schedule to CSMS as a NotifyEVChargingScheduleRequest.
	Scenario description #2	Via EV when using ISO 15118-20 Dynamic Control mode 1. EV sends an AC/DC_ChargeLoopReq message with new DepartureTime and/or EVTargetEnergyRequest. 2. CS sends a NotifyEVChargingNeedsRequest with given *departureTime* and *evTargetEnergyRequest* to CSMS. 3. CSMS calculates a new charging profile and sends a SetChargingProfileRequest to the Charging Station. 4. CS controls charging of EV via the AC/DC_ChargeLoopRes message to match the provided charging profile.
	Scenario description #3	Via charging station when using ISO 15118-20 Dynamic Control mode 1. CS respond to an AC/DC_ChargeLoopReq message with an AC/DC_ChargeLoopRes message with a new DepartureTime. 2. CS sends a NotifyEVChargingNeedsRequest with given *departureTime* and *evTargetEnergyRequest* to CSMS. 3. CSMS calculates a new charging profile and sends a SetChargingProfileRequest to the Charging Station. 4. CS controls charging of EV via the AC/DC_ChargeLoopRes message to match the provided charging profile.
5	**Prerequisites**	A charging session based on ISO 15118-20 is already active.
6	**Post conditions**	EV is charging according to an updated charging profile.
7	**Error Handling**	

No.	Type	Description
8	Remarks	


EV
Charging Station
CSMS
Transaction is in progress.
Departure time or energy needs have changed
alt
[15118-20 Scheduled mode]
ScheduleExchangeReq(NewDepartureTime, NewEVTargetEnergyRequest, ...)
NotifyEVChargingNeedsRequest(NewDepartureTime, NewEVTargetEnergyRequest, ...)
NotifyEVChargingNeedsResponse(Accepted)
calculate updated charging profile
SetChargingProfileRequest(v2xSchedule)
SetChargingProfileResponse(Accepted)
ScheduleExchangeRes(Schedule)
PowerDeliveryReq(SelectedScheduleTupleID, EVPowerProfile)
PowerDeliveryRes()
NotifyEVChargingScheduleRequest(EVChargingSchedule)
NotifyEVChargingScheduleResponse (Accepted)
EV follows schedule
[15118-20 Dynamic mode]
opt
[Departure time and/or energy need changed by EV]
EV sends new time/energy
AC/DC_ChargeLoopReq(NewDepartureTime, NewEVTargetEnergyRequest, ...)
AC/DC_ChargeLoopRes(EVSETargetActivePower, ...)
[Departure time and/or energy need changed by charging station]
Change of time/energy by user at charging station
AC/DC_ChargeLoopReq(...)
AC/DC_ChargeLoopRes(NewDepartureTime, NewEVTargetEnergyRequest, ...)
EV confirms new time/energy
AC/DC_ChargeLoopReq(NewDepartureTime, NewEVTargetEnergyRequest, ...)
AC/DC_ChargeLoopRes(EVSETargetActivePower, ...)
NotifyEVChargingNeedsRequest(DepartureTime, EVTargetEnergyRequest, ...)
NotifyEVChargingNeedsResponse(Accepted)
Charge loop continues
calculate updated charging profile
SetChargingProfileRequest(v2xSchedule)
SetChargingProfileResponse(Accepted)
Adjust power to setpoint in period
AC/DC_ChargeLoopReq()
AC/DC_ChargeLoopRes(EVSETargetActivePower, ...)
[CHAdeMO]
Charging needs are not communicated by CHAdeMO

K20 - ISO 15118-20 Adjusting charging schedule when energy needs change - Requirements

Table 116. K20 - Requirements

ID	Precondition	Requirements	Note
K20.FR.01	Charging Station receives a new departure time and/or target energy amount from EV	Charging Station sends a NotifyEVChargingNeedsRequest with a *chargingNeeds* element with new *departureTime* and/or a *v2xChargingParameters* element with new *evTargetEnergyRequest* to CSMS	
K20.FR.02	K20.FR.01 AND EV and Charging Station use ISO 15118-20 "Scheduled Control Mode"	Behavior is as described in K17 - Renegotiation intiated by EV	
K20.FR.03	K20.FR.01 AND EV and Charging Station use ISO 15118-20 "Dynamic Control Mode"	CSMS calculates a new charging profile and sends a SetChargingProfileRequest to Charging Station	
K20.FR.04	K20.FR.03	Charging Stations controls charging of EV via the AC/DC_ChargeLoopRes message to match the given charging profile *limit* or *setpoint*.	
K20.FR.05	K20.FR.01 AND Charging Station is offline	Charging Station SHOULD add *timestamp* to the NotifyEVChargingNeedsRequest with the time when charging needs were received from EV	This will tell CSMS how old this data is, if it was not immediately sent because of an offline period.

5.4. Priority Charging

K21 - Requesting priority charging remotely

New in OCPP 2.1

No.	Type	Description
1	**Name**	Requesting priority charging remotely
2	**ID**	K21
3	**Objective(s)**	To let a user request immediate charging via CSMS (e.g. via smartphone app)
4	**Description**	The user requests CSMS to charge the EV immediately and avoid V2X discharging. CSMS instructs CS to charge the EV with the highest power possible under the circumstances. Note, that his may be less than the maximum capacity of the CS when other restrictions (e.g. load-balancing) are in effect.
	Actors	Charging Station, CSMS
	Scenario description	1. The CSMS is triggered to request priority charging for the active transaction at the EVSE (how this is done is outside scope of OCPP). 2. The CSMS sends a UsePriorityChargingRequest message with *activate* = true for transaction *transactionId*. 3. If Charging Station has an `PriorityCharging` profile, then it will respond with UsePriorityChargingResponse `Accepted`. 4. The Charging Station stops applying the `TxDefaultProfile` or `TxProfile` charging profile (if active) to this transaction, and applies the `PriorityCharging` profile instead. 5. The Charging Station responds with a NotifyPriorityChargingRequest with *activated* = true and *transactionId* set to the transaction to confirm that it now applies the instant charging profile. 6. The CSMS confirms receipt with a NotifyPriorityChargingResponse.
5	**Prerequisites**	CSMS has sent a charging profile with *chargingProfilePurpose* = `PriorityCharging` to the Charging Station for this EVSE or for EVSE #0. The Charging Station is online and a transaction is active.
6	**Post conditions**	The transaction continues using the `PriorityCharging` profile.

No.	Type	Description
7	**Error Handling**	If Charging Station has no `PriorityCharging` profile, then it responds with `NoProfile`. If it cannot activate for other reasons, (e.g. because *transactionId* does not exist) then it responds with `Rejected`. In both cases, the use case ends.
8	**Remarks**	The priority charging profile remains in effect for the duration of the transaction. It ceases to be active when the charging session is terminated, or when CSMS requests priority charging to stop. The use of priority charging is not restricted to V2X operations only. A priority charging profile can also be requested for non-V2X transactions. A priority charging profile can be deactivated by sending a NotifyPriorityChargingRequest with *activated* = false. It is possible to provide a PriorityCharging profile with a *maxOfflineDuration* and a high stack level and another PriorityChargingProfile with no *maxOfflineDuration* and a lower stack level. The first one is then only used as long as the charging station is online and can therefore utilize higher power levels. The second profile becomes active when the charging station goes offline and utilizes safer lower power levels.

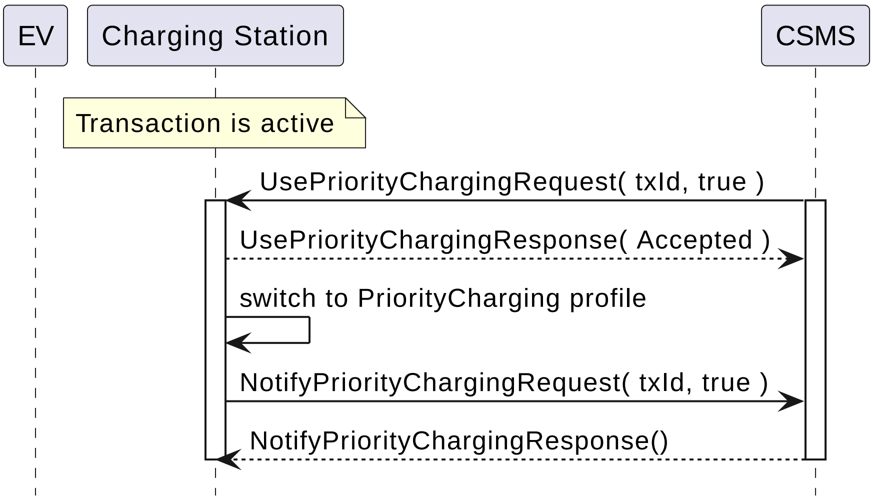

EV
Charging Station
CSMS
Transaction is active
UsePriorityChargingRequest(txId, true)
UsePriorityChargingResponse(Accepted)
switch to PriorityCharging profile
NotifyPriorityChargingRequest(txId, true)
NotifyPriorityChargingResponse()

Figure 144. Requesting PriorityCharging remotely

K21 - Requesting priority charging remotely - Requirements

Table 117. K21 - Requirements

ID.	Precondition	Requirements	Note
K21.FR.01	Charging Station receives a UsePriorityChargingRequest with *transactionId* that matches an ongoing transaction AND Charging Station does not have a `PriorityCharging` charging profile installed on EVSE #0 or the EVSE of the transaction	Charging Station SHALL respond with UsePriorityChargingResponse with *status* = `NoProfile` and an optional *reasonCode* = "NotFound"	A `PriorityCharging` profile on EVSE #0 applies to all EVSEs.
K21.FR.02	Charging Station receives a UsePriorityChargingRequest with *transactionId* that does not match an ongoing transaction	Charging Station SHALL respond with UsePriorityChargingResponse with *status* = `Rejected` and an optional *reasonCode* = "TxNotFound"	
K21.FR.03	Charging Station receives a UsePriorityChargingRequest with *transactionId* that matches an ongoing transaction AND Charging Station has a `PriorityCharging` charging profile installed on EVSE #0 or the EVSE of the transaction	Charging Station SHALL respond with UsePriorityChargingResponse with *status* = `Accepted`	

K21.FR.04	K21.FR.03	Charging Station SHALL stop applying the `TxDefaultProfile` or `TxProfile` charging profile (if active) to this transaction and applies the `PriorityCharging` profile instead.	`ChargingStationMaxProfile` charging profiles remains in effect.
K21.FR.05	K21.FR.04	Charging Station SHALL send a NotifyPriorityChargingRequest with *activated* = true and *transactionId* set to the associated transaction	
K21.FR.06	Charging Station receives a UsePriorityChargingRequest with *transactionId* that matches an ongoing transaction and *activate* = false	Charging Station SHALL respond with UsePriorityChargingResponse with *status* = `Accepted`	If no `PriorityCharging` profile was active, then this has no effect.
K21.FR.07	K21.FR.06 AND `PriorityCharging` is active for the transaction	Charging Station SHALL stop applying the `PriorityCharging` profile and reapply the `TxDefaultProfile` or `TxProfile` charging profile (if existing) to this transaction.	
K21.FR.08	K21.FR.07	Charging Station SHALL send a NotifyPriorityChargingRequest with *activated* = false and *transactionId* set to the associated transaction	
K21.FR.09	K21.FR.05 OR K21.FR.08	CSMS SHALL respond with NotifyPriorityChargingResponse.	

K22 - Requesting priority charging locally

New in OCPP 2.1

No.	Type	Description
1	**Name**	Requesting priority charging locally
2	**ID**	K22
3	**Objective(s)**	To let a user request immediate charging at the Charging Station.
4	**Description**	The user requests to charge the EV immediately and avoid V2X discharging. Charging Station switches to a priority charging schedule to charge the EV with the highest power possible under the circumstances. Note, that this may be less than the maximum capacity of the CS when other restrictions (e.g. load-balancing) are in effect.
	Actors	Charging Station, CSMS
	Scenario description	1. User instructs Charging Station (via button or display) to request a priority charging for the transaction. 2. The Charging Station stops applying the `TxDefaultProfile` or `TxProfile` charging profile (if active) to this transaction, and applies the `PriorityCharging` profile instead. 3. The Charging Station responds with a NotifyPriorityChargingRequest with *activated* = true and *transactionId* set to the transaction to tell CSMS that it now applies the priority charging profile. 4. The CSMS confirms receipt with a NotifyPriorityChargingResponse.
5	**Prerequisites**	The Charging Station is online and a transaction is active.
6	**Post conditions**	Transaction using the priority charging profile with the highest possible power profile that CSMS deems possible under the circumstances.
7	**Error Handling**	If Charging Station has no `PriorityCharging` profile, then it cannot switch to priority charging.
8	**Remarks**	If the Charging Station is offline, it switches to the `PriorityCharging` profile and queues the NotifyPriorityChargingRequest message to be transmitted when it is back online. The priority charging profile remains in effect for the duration of the transaction. It ceases to be active when the charging session is terminated, or when priority charging is requested to stop. The use of priority charging is not restricted to V2X operations only. A priority charging profile can also be requested for non-V2X transactions.

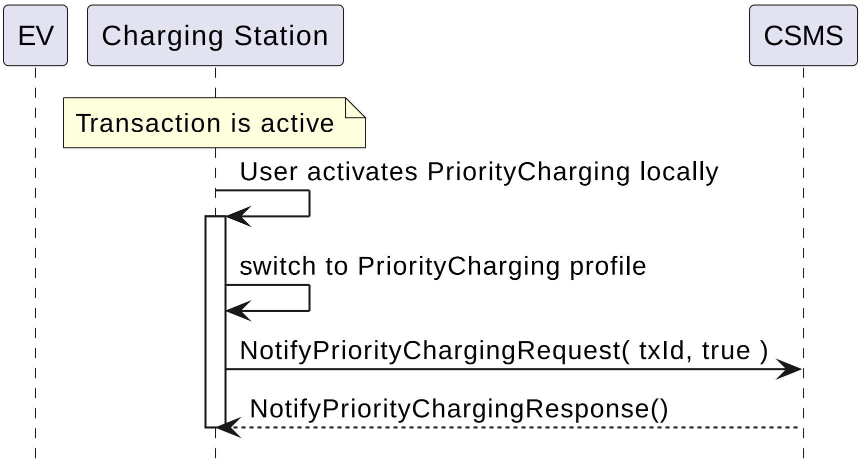

EV
Charging Station
CSMS
Transaction is active
User activates PriorityCharging locally
switch to PriorityCharging profile
NotifyPriorityChargingRequest(txId, true)
NotifyPriorityChargingResponse()

Figure 145. Requesting PriorityCharging locally

K22 - Requesting priority charging locally - Requirements

Table 118. K22 - Requirements

ID.	Precondition	Requirements	Note
K22.FR.01	User instructs Charging Station (via button or display) to request a priority charging for the transaction AND No `PriorityCharging` profile is present, that is valid for the associated EVSE	Charging Station SHALL notify user it is not capable of `PriorityCharging`.	
K22.FR.02	User instructs Charging Station (via button or display) to request a priority charging for the transaction AND a `PriorityCharging` profile, that is valid for the associated EVSE, has been installed earlier.	Charging Station SHALL stop applying the `TxDefaultProfile` or `TxProfile` charging profile (if active) to this transaction and apply the `PriorityCharging` profile instead	`ChargingStationMaxProfile` charging profile remains in effect.
K22.FR.03	K22.FR.02	Charging Station SHALL send a NotifyPriorityChargingRequest with *activated* = true and *transactionId* set to the affected transaction	
K22.FR.04	User instructs Charging Station (via button or display) to end priority charging for the transaction	Charging Station SHALL stop applying the `PriorityCharging` charging profile and reapply the `TxDefaultProfile` or `TxProfile` charging profile (if active) to this transaction instead.	Charging Station UI should not allow this option when priority charging was not in effect.
K22.FR.05	K22.FR.04	Charging Station SHALL send a NotifyPriorityChargingRequest with *activated* = false and *transactionId* set to the affected transaction	
K22.FR.06	K22.FR.03 OR K22.FR.05	CSMS SHALL respond with NotifyPriorityChargingResponse.	

5.5. Dynamic Charging Profiles

Dynamic charging profiles are a way to provide dynamic charging limit updates without a charging schedule. The mechanism of a charging profile with a charging schedule is still used, but the schedule only consists of a single period for which the *limits* or *setpoints* can be changed dynamically via the UpdateDynamicScheduleRequest or PullDynamicScheduleUpdateResponse message from CSMS.

Dynamic charging profiles are useful for changes in (dis)charging limits that cannot be scheduled, because of their unpredictable nature. It matches well with the ISO 15118-20 "dynamic control mode", which does not use a schedule either.

The use of dynamic charging profiles is suboptimal in combination with ISO 15118-20 "scheduled control mode" or ISO 15118-2, which both rely on charging schedules. Any change in a limit or setpoint of the dynamic charging profile will cause a renegotiation between EV and Charging Station for a new charging schedule.

K28 - Dynamic charging profiles from CSMS

New in OCPP 2.1

No.	Type	Description
1	**Name**	Dynamic charging profiles from CSMS
2	**ID**	K28
3	**Objective(s)**	To allow a dynamically changing limit in a charging profile.
4	**Description**	A dynamic ChargingProfileType consists of a single schedule period in which the *limit* or *setpoint* can be updated without having to replace the charging profile. The updated value is either sent by CSMS or pulled by Charging Station via messages.
	Actors	Charging Station, CSMS, external system (EMS)
	Scenario description #1	*Updates sent by CSMS* 1. CSMS sends a SetChargingProfileRequest with a ChargingProfileType with *chargingProfileKind* = `Dynamic` and a single *chargingSchedule* with one *chargingSchedulePeriod* with a *limit* and without a *dynUpdateInterval*. 2. Charging Station responds with SetChargingProfileResponse with *status* = `Accepted` and applies *limit* to charging, and sets *dynUpdateTime* to current time. 3. When CSMS wishes to update the *limit* it sends a UpdateDynamicScheduleRequest with the *chargingProfileId* of the charging profile and a ChargingScheduleUpdateType with a new value of *limit*. 4. Charging stations responds with UpdateDynamicScheduleResponse with *status* = `Accepted`, sets *dynUpdateTime* in ChargingProfileType to current time, and applies the new *limit*.
	Scenario description #2	*Updates requested by Charging Station* 1. CSMS sends a SetChargingProfileRequest with a ChargingProfileType with *chargingProfileKind* = `Dynamic` and a single *chargingSchedule* with one *chargingSchedulePeriod* with a *limit* and with a *dynUpdateInterval* of 60 seconds, which tells Charging Station to pull for updates every 60 seconds. 2. Charging Station responds with SetChargingProfileResponse with *status* = `Accepted` and applies *limit* to charging, and sets *dynUpdateTime* to current time. 3. When the time is *dynUpdateTime* + *dynUpdateInterval* Charging Station requests an update by sending a PullDynamicScheduleUpdateRequest with the *chargingProfileId* of the charging profile. 4. CSMS reponds with PullDynamicScheduleUpdateResponse with *status* = `Accepted` and a ChargingScheduleUpdateType with a new value for *limit*. 5. Charging Station sets *dynUpdateTime* in ChargingProfileType to current time, and applies the new *limit*.
5	**Prerequisites**	Charging Station supports Dynamic charging profiles.
6	**Post conditions**	
7	**Error Handling**	
8	**Remarks**	

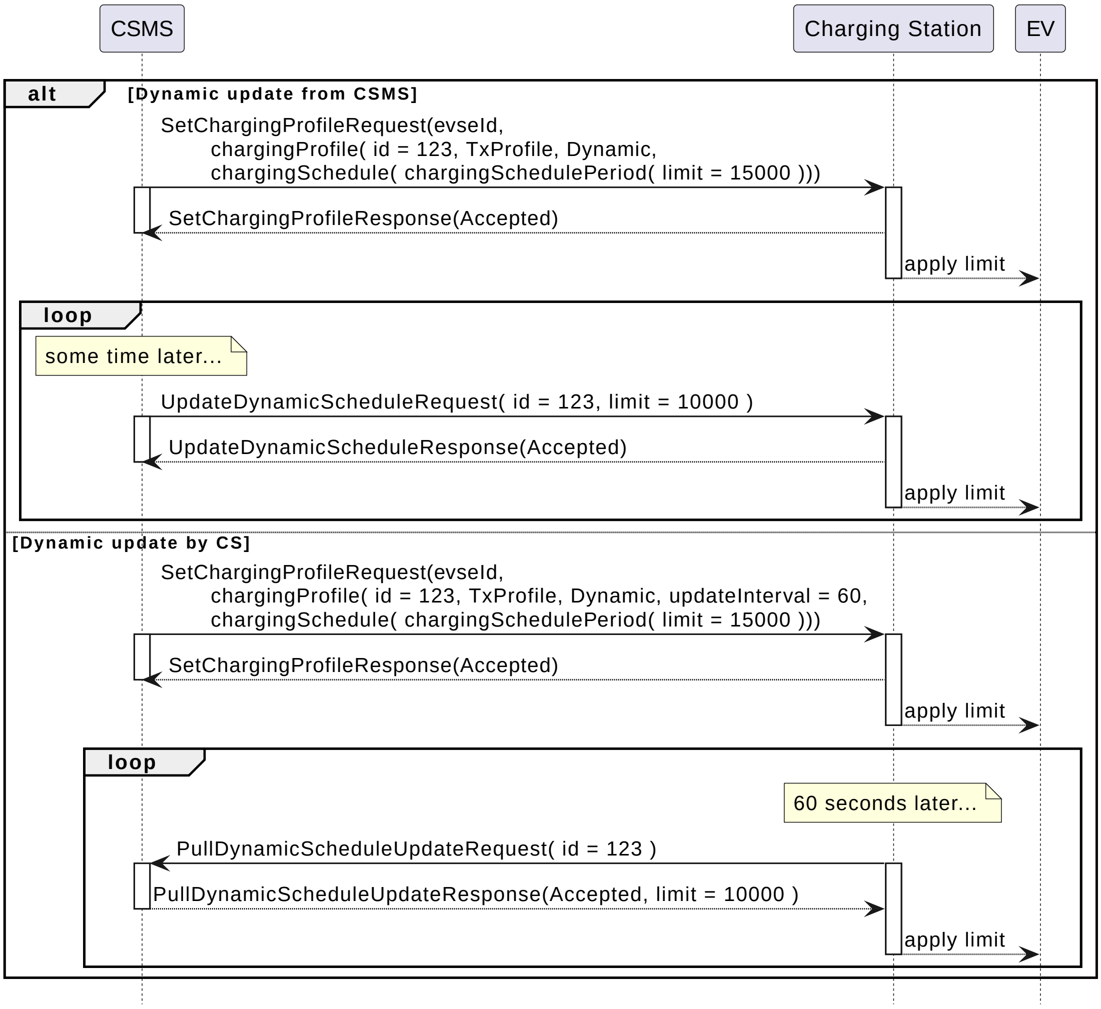

CSMS
Charging Station
EV
alt
[Dynamic update from CSMS]
SetChargingProfileRequest(evseId,
chargingProfile(id = 123, TxProfile, Dynamic,
chargingSchedule(chargingSchedulePeriod(limit = 15000)))
SetChargingProfileResponse(Accepted)
apply limit
loop
some time later...
UpdateDynamicScheduleRequest(id = 123, limit = 10000)
UpdateDynamicScheduleResponse(Accepted)
apply limit
[Dynamic update by CS]
SetChargingProfileRequest(evseId,
chargingProfile(id = 123, TxProfile, Dynamic, updateInterval = 60,
chargingSchedule(chargingSchedulePeriod(limit = 15000)))
SetChargingProfileResponse(Accepted)
apply limit
loop
60 seconds later...
PullDynamicScheduleUpdateRequest(id = 123)
PullDynamicScheduleUpdateResponse(Accepted, limit = 10000)
apply limit

Figure 146. Sequence Diagram: Dynamic charging profile

K28 - Dynamic charging profiles from CSMS - Requirements

Table 119. K28 - Requirements

ID.	**Precondition**	**Requirements**	**Note**
K28.FR.01		A ChargingProfileType with *chargingProfileKind* = `Dynamic` SHALL consist of only 1 ChargingScheduleType with only 1 ChargingSchedulePeriodType	
K28.FR.02		The ChargingSchedulePeriodType in a ChargingProfileType with *chargingProfileKind* = `Dynamic` SHALL have a *startPeriod* = 0	It becomes immediately active upon receipt of the message.
K28.FR.03	When Charging Station receives a SetChargingProfileRequest in which *chargingSchedule* does not obey K28.FR.01 or K28.FR.02	Charging Station SHALL respond with SetChargingProfileResponse with *status* = `Rejected` and optionally a *statusInfo.reasonCode* = "InvalidSchedule"	
K28.FR.04	When Charging Station receives a SetChargingProfileRequest with a *chargingProfileKind* that is not `Dynamic` and with a field *dynUpdateInterval*	Charging Station SHALL respond with SetChargingProfileResponse with *status* = `Rejected` and optionally a *statusInfo.reasonCode* = "InvalidProfile".	*dynUpdateInterval* only applies to Dynamic profiles.
K28.FR.05	When Charging Station receives a SetChargingProfileRequest with *chargingProfileKind* = `Dynamic` AND *dynUpdateTime* is not set	Charging Station SHALL set *dynUpdateTime* in ChargingProfileType to current time.	

K28.FR.06	When Charging Station receives a UpdateDynamicScheduleRequest for a charging profile that has *chargingProfileKind* = `Dynamic`	Charging Station SHALL respond with *status* = `Accepted` and immediately apply the values from UpdateDynamicScheduleRequest to the charging profile.	
K28.FR.07	When CSMS receives a PullDynamicScheduleUpdateRequest for a charging profile that has *chargingProfileKind* = `Dynamic`	CSMS SHALL respond with PullDynamicScheduleUpdateResponse with *status* = `Accepted` and the limits/setpoints to use in the charging profile.	
K28.FR.08	When Charging Station receives a PullDynamicScheduleUpdateResponse for a charging profile that has *chargingProfileKind* = `Dynamic`	Charging Station SHALL immediately apply the values from PullDynamicScheduleUpdateResponse to the charging profile.	
K28.FR.09	K28.FR.06 OR K28.FR.07 OR K28.FR.08	Charging Station SHALL set *dynUpdateTime* to current time.	
K28.FR.10	When *chargingProfileKind* = `Dynamic` and *dynUpdateInterval* > 0 in *chargingProfile*	Charging Station SHALL send a PullDynamicScheduleUpdateRequest with *chargingProfileId* = *chargingProfile.id* to request an update of the *chargingSchedulePeriod*.	
K28.FR.11	When Charging Station receives a UpdateDynamicScheduleRequest for a charging profile that does not have *chargingProfileKind* = `Dynamic`	Charging Station SHALL respond with a *status* = `Rejected` and optionally a *statusInfo.reasonCode* = `InvalidProfile`	
K28.FR.12	When CSMS receives a PullDynamicScheduleUpdateRequest for a charging profile that does not exist or does not have *chargingProfileKind* = `Dynamic`	CSMS SHALL respond with a *status* = `Rejected` and optionally a *statusInfo.reasonCode* = `InvalidProfile`	

K29 - Dynamic charging profiles from external system

New in OCPP 2.1

No.	**Type**	**Description**
1	**Name**	Dynamic charging profiles from external system
2	**ID**	K29
3	**Objective(s)**	To allow a dynamically changing limit in a charging profile from an external system.
4	**Description**	A dynamic ChargingProfileType consists of a single schedule period in which the *limit* or *setpoint* can be updated without having to replace the charging profile. *operationMode* is set to `ExternalLimits` or `ExternalSetpoint`, which means that the *limit* or *setpoint* is set directly by an external system, e.g. an energy management system. This can be a charging limit or schedule that is received from an energy management system (via any interface) and which is represented by Charging Station as a charging profile of purpose `ChargingStationExternalConstraints`. If `ExternalConstraintsProfileDisallowed` is true, then an external system is not allowed to set a `ChargingStationExternalConstraints` profile. In that case CSMS decides when the limits from the external system are taken into account by submitting a charging profile of purpose `TxProfile` or `TxDefaultProfile` with a *chargingSchedulePerioc* that has *operationMode* = `ExternalLimits` or `ExternalSetpoint`.
	Actors	Charging Station, CSMS, external system (EMS)

No.	Type	Description
	Scenario description #1	*Charging profile from external system with dynamic updates* 1. The external system provides limits, which the Charging Station internally represents as a ChargingProfileType with *chargingProfilePurpose* = `ChargingStationExternalConstraints`, *chargingProfileKind* = `Dynamic` and a single *chargingSchedule* with one *chargingSchedulePeriod* with an *operationMode* = `ExternalLimits` (or `ExternalSetpoint`) 2. Charging Station applies *limit* (or *setpoint*) to charging. 3. Whenever the external system updates the *limit* (or *setpoint*), Charging Station applies the new *limit* (or *setpoint*). 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, as described in use case K11.
	Scenario description #2	*Charging profile from CSMS with dynamic updates from external system* 1. CSMS sends a SetChargingProfileRequest with a ChargingProfileType with *chargingProfilePurpose* = `TxProfile` (or `TxDefaultProfile`), *chargingProfileKind* = `Dynamic` and a single *chargingSchedule* with one *chargingSchedulePeriod* with an *operationMode* = `ExternalLimits` (or `ExternalSetpoint`), which tells Charging Station that an external system (e.g. an EMS) will provide updates to the *limit* (or *setpoint) to use in this _chargingSchedulePeriod*. 2. Charging Station responds with SetChargingProfileResponse with *status* = `Accepted` and applies *limit* to charging. 3. Whenever the external system updates the *limit* (or *setpoint*), Charging Station applies the new *limit* (or *setpoint*). 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, as described in use case K11.
5	**Prerequisites**	Charging Station supports Dynamic charging profiles, and is able to receive power limits from an external system.
6	**Post conditions**	External system can dynamically update the power limit in the charging profile.
7	**Error Handling**	
8	**Remarks**	When an EMS provides frequent updates to a power limit, instead of providing a schedule, then each update would result in Charging Station having to replace the existing `ChargingStationExternalConstraints` charging profile by a new one. When using a `Dynamic` charging profile Charging Station only has to update the value of *limit* or *setpoint* of the existing charging profile.

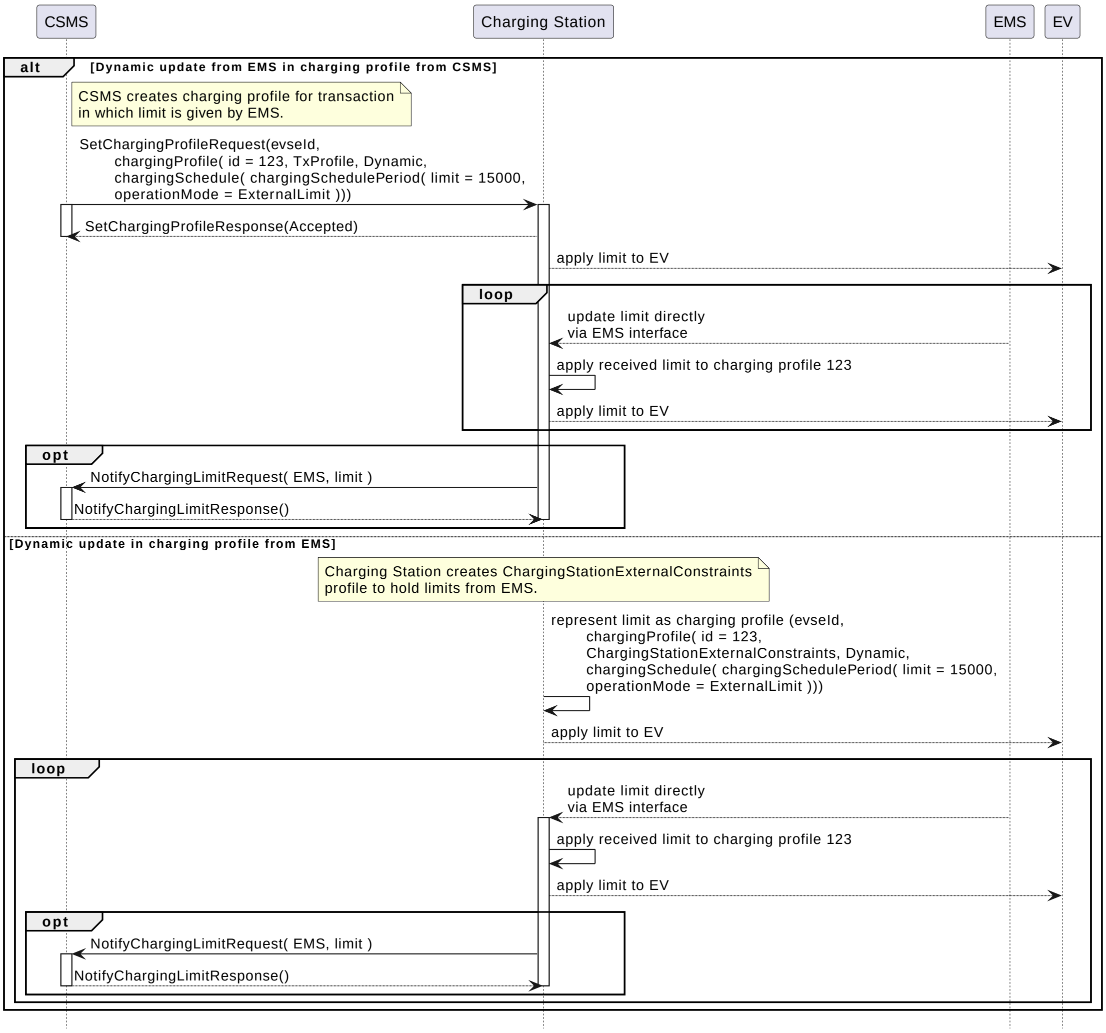

CSMS
Charging Station
EMS
EV
alt
[Dynamic update from EMS in charging profile from CSMS]
CSMS creates charging profile for transaction in which limit is given by EMS.
SetChargingProfileRequest(evseId, chargingProfile(id = 123, TxProfile, Dynamic, chargingSchedule(chargingSchedulePeriod(limit = 15000, operationMode = ExternalLimit)))
SetChargingProfileResponse(Accepted)
apply limit to EV
loop
update limit directly via EMS interface
apply received limit to charging profile 123
apply limit to EV
opt
NotifyChargingLimitRequest(EMS, limit)
NotifyChargingLimitResponse()
[Dynamic update in charging profile from EMS]
Charging Station creates ChargingStationExternalConstraints profile to hold limits from EMS.
represent limit as charging profile (evseId, chargingProfile(id = 123, ChargingStationExternalConstraints, Dynamic, chargingSchedule(chargingSchedulePeriod(limit = 15000, operationMode = ExternalLimit)))
apply limit to EV
loop
update limit directly via EMS interface
apply received limit to charging profile 123
apply limit to EV
opt
NotifyChargingLimitRequest(EMS, limit)
NotifyChargingLimitResponse()

Figure 147. Sequence Diagram: Dynamic charging profile

K29 - Dynamic charging profiles by external system - Requirements

Table 120. K29 - Requirements

ID.	Precondition	Requirements	Note
K29.FR.01		A ChargingProfileType with *chargingProfileKind* = `Dynamic` SHALL consist of only 1 ChargingScheduleType with only 1 ChargingSchedulePeriodType	(Same as K28.FR.01)
K29.FR.02		The ChargingSchedulePeriodType in a ChargingProfileType with *chargingProfileKind* = `Dynamic` SHALL have a *startPeriod* = 0	It becomes immediately active upon receipt of the message. (Same as K28.FR.02)
K29.FR.03	When Charging Station receives a SetChargingProfileRequest from CSMS with a ChargingProfileType with *chargingProfilePurpose* = `TxProfile` or `TxDefaultProfile`, *chargingProfileKind* = `Dynamic` and *operationMode* = `ExternalLimits` or `ExternalSetpoint`	Charging Station SHALL represent the limit or setpoint that it receives from an external system as the *limit* or *setpoint* of this charging profile.	
K29.FR.04	NOT K29.FR.03 AND `ExternalConstraintsProfileDisallowed` is false or absent AND An external system provides a current or power limit (i.e. single value, not a schedule)	Charging Station SHALL represent this as a ChargingProfileType with a single *chargingSchedulePeriod*, and having a *chargingProfilePurpose* = `ChargingStationExternalConstraints` with a *chargingProfileKind* = `Dynamic`.	The alternative, using a *chargingProfileKind* = `Absolute`, is described in K11.FR.06.

K29.FR.05	When external system updates a limit AND Charging Station represents this as a `Dynamic` charging profile	Charging Station SHALL update the *limit* or *setpoint* in this charging profile.	
K29.FR.06	K29.FR.05	Charging Station SHALL set *dynUpdateTime* in ChargingProfileType to current time.	

L. Firmware Management

Chapter 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.

Chapter 2. Use cases & Requirements

L01 - Secure Firmware Update

No.	Type	Description
1	**Name**	Secure Firmware Update
2	**ID**	L01
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
	Scenario description	**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. **2.** The Charging Station verifies the validity of the certificate against the Manufacturer root certificate. **3.** If the certificate is valid AND the retrieveDateTime has passed, 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). **4.** If the Firmware successfully downloaded, the Charging Station sends a FirmwareStatusNotificationRequest with status Downloaded. Otherwise, it sends a FirmwareStatusNotificationRequest with status DownloadFailed. **5.** If the verification is successful AND the installDateTime has passed, 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). **6.** If the installation is successful, the Charging Station sends a FirmwareStatusNotificationRequest with status Installed. Otherwise, it sends a FirmwareStatusNotificationRequest with status InstallationFailed.
	Alternative scenario(s)	L02 - Non-Secure Firmware Update
5	**Prerequisite(s)**	The Charging Station Manufacturer provided a firmware update.
6	**Postcondition(s)**	**Successful postcondition:** The firmware is updated and the Charging Station is in *Installed* status. **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.
7	**Error handling**	n/a

No.	Type	Description
8	**Remark(s)**	As an example in this use case the requestId = 123, but this could be any value. Measures SHOULD be taken to secure the firmware when it is stored on a server or workstation. The Charging Station has a required Configuration Variable that reports which file transfer protocols it supports: `FileTransferProtocols` When migrating to a new version of OCPP it is RECOMMENDED to install a fallback NetworkConnectionProfile with the new configuration. The requirements for the Firmware Signing Certificate are described in the: Certificate Properties section. The manufacturer SHALL NOT use intermediate certificates for the firmware signing certificate in the Charging Station. FTP needs to be able to use Passive FTP, to be able to transverse over as many different typologies as possible.

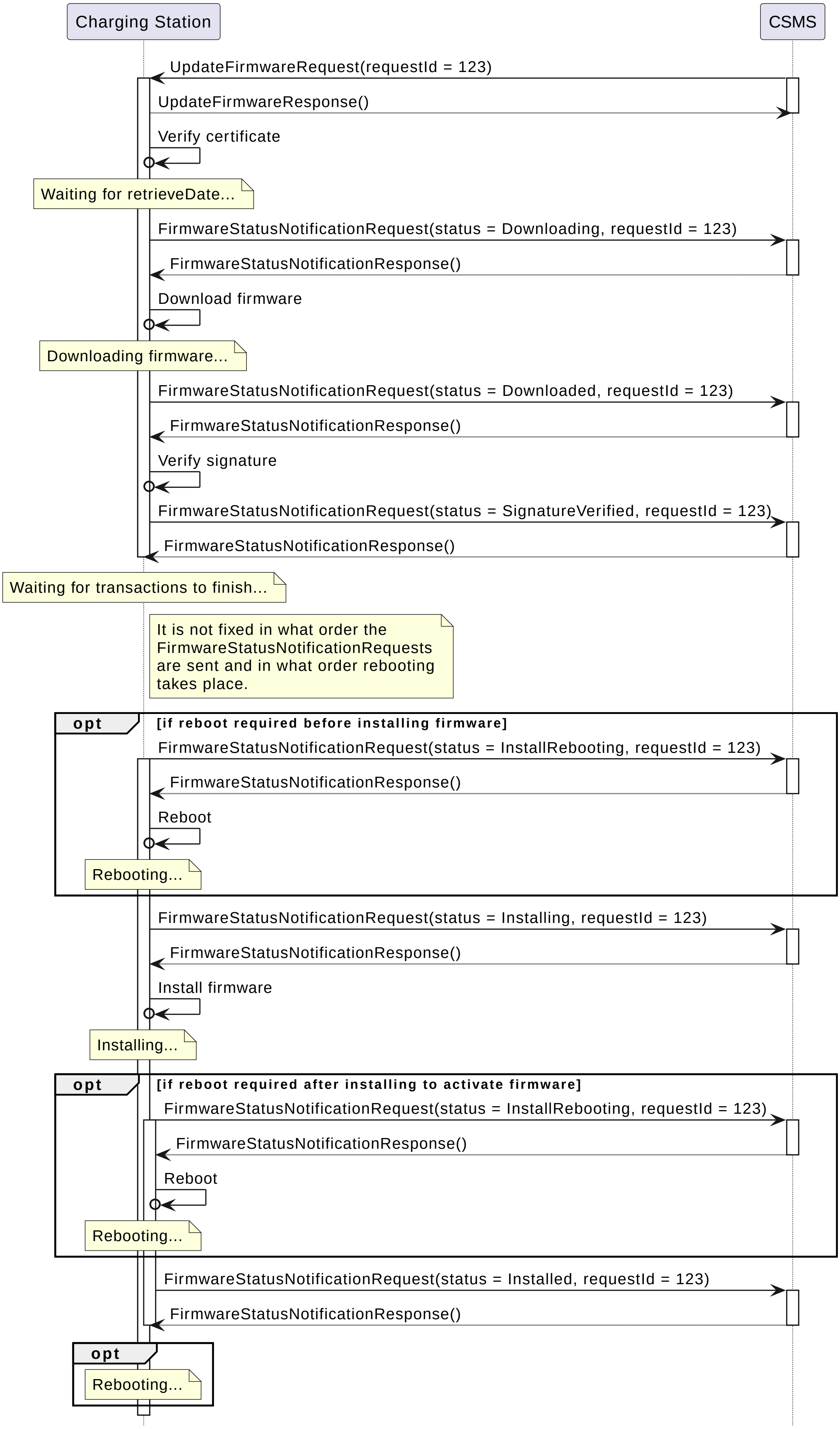

Charging Station
CSMS
UpdateFirmwareRequest(requestId = 123)
UpdateFirmwareResponse()
Verify certificate
Waiting for retrieveDate...
FirmwareStatusNotificationRequest(status = Downloading, requestId = 123)
FirmwareStatusNotificationResponse()
Download firmware
Downloading firmware...
FirmwareStatusNotificationRequest(status = Downloaded, requestId = 123)
FirmwareStatusNotificationResponse()
Verify signature
FirmwareStatusNotificationRequest(status = SignatureVerified, requestId = 123)
FirmwareStatusNotificationResponse()
Waiting for transactions to finish...
It is not fixed in what order the FirmwareStatusNotificationRequests are sent and in what order rebooting takes place.
opt
[if reboot required before installing firmware]
FirmwareStatusNotificationRequest(status = InstallRebooting, requestId = 123)
FirmwareStatusNotificationResponse()
Reboot
Rebooting...
FirmwareStatusNotificationRequest(status = Installing, requestId = 123)
FirmwareStatusNotificationResponse()
Install firmware
Installing...
opt
[if reboot required after installing to activate firmware]
FirmwareStatusNotificationRequest(status = InstallRebooting, requestId = 123)
FirmwareStatusNotificationResponse()
Reboot
Rebooting...
FirmwareStatusNotificationRequest(status = Installed, requestId = 123)
FirmwareStatusNotificationResponse()
opt
Rebooting...

Figure 148. Sequence diagram secure firmware upgrade (happy flow)

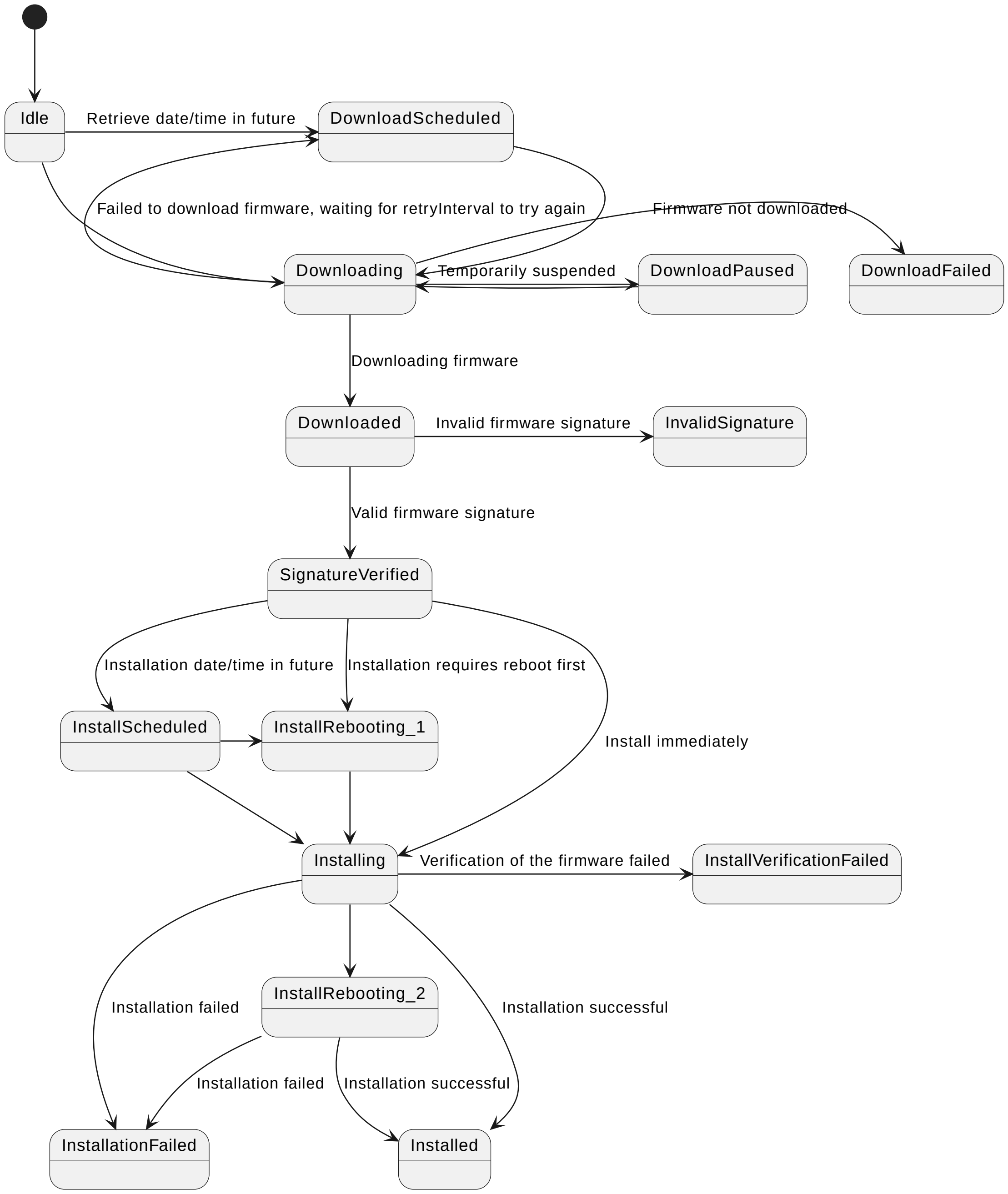


Figure 149. Firmware status transitions

L01 - Secure Firmware Update - Requirements

Table 121. 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 ECSchnorr algorithm for signing, and the SHA256 algorithm for calculating hash values.	
L01.FR.05	L01.FR.04 AND (*installDateTime* is not set OR current time >= *installDateTime*)	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 start installation of firmware while a transaction is ongoing	The Charging Station SHALL wait until all transactions have ended, before commencing installation.	
L01.FR.07	L01.FR.06 or L01.FR.33 AND configuration variable `AllowNewSessionsPendingFirmwareUpdate` is *false* or does not exist	The Charging Station SHALL set all EVSE that are not in use to UNAVAILABLE while the Charging Station waits for the ongoing transactions to end. Until the firmware is installed, any EVSE 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.	

ID	Precondition	Requirement definition	Note
L01.FR.13	When the Charging Station does not start downloading firmware, because it is busy charging or because *retrieveDateTime* 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.
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 *installDateTime* 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 *requestId* in FirmwareStatusNotificationRequest is mandatory, unless *status* = 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. The Charging Station may send a FirmwareStatusNotificationRequest with *status* `DownloadFailed` or `InstallationFailed` for the firmware update that has now been canceled.
L01.FR.25	Charging Station receives a TriggerMessageRequest for `FirmwareStatusNotification` AND last sent FirmwareStatusNotificationRequest had *status* = Installed	Charging Station SHALL return a FirmwareStatusNotificationRequest with *status* = Idle.	
L01.FR.26	Charging Station receives a TriggerMessageRequest for `FirmwareStatusNotification` AND last sent FirmwareStatusNotificationRequest had NOT *status* Installed	Charging Station SHALL return a FirmwareStatusNotificationRequest with the last sent *status*.	
L01.FR.27	L01.FR.24 AND the Charging Station is unable to cancel the firmware installation	The Charging Station MAY respond with *status* = `Rejected`.	

ID	Precondition	Requirement definition	Note
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 *type* = "FirmwareUpdated".	
L01.FR.32	When a Charging Station has successfully installed the new firmware AND the Charging Station needs to reboot before activating the new firmware	The Charging Station SHALL either: (a) send an optional FirmwareStatusNotificationRequest with *status* = `InstallRebooting` before rebooting and send a mandatory FirmwareStatusNotificationRequest with *status* = `Installed` by the newly activated firmware, or (b) only send a FirmwareStatusNotificationRequest with status set to `Installed` without reporting the reboot and activation of the new firmware.	Option (a) is preferred, because it notifies CSMS of an upcoming reboot of the Charging Station, and the final *status* = `Installed` is sent by the new firmware image, so that CSMS can be sure that the new firmware is active. This is not guaranteed by option (b) when rebooting of the new firmware should fail.
L01.FR.33 **(2.1)**	L01.FR.05 AND The Charging Station has ongoing transactions AND a reboot is needed to activate the installed firmware	The Charging Station SHALL wait until all transactions have ended, before activating the installed firmware.	E.g. in case of A/B firmware updates. To notify the CSMS when the new firmware is active it is recommended to follow the flow of option (a) in L01.FR.32.
L01.FR.34	L01.FR.04 AND *installDateTime* is not set AND Charging Station is waiting for a transaction to finish	The Charging Station MAY send a FirmwareStatusNotificationRequest with status InstallScheduled.	The case where *installDateTime* is set is covered by L01.FR.16.

L02 - Non-Secure Firmware Update

No.	Type	Description
1	**Name**	Non-Secure Firmware Update
2	**ID**	L02
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

No.	Type	Description
	Scenario description	**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. **2.** The Charging station responds with an UpdateFirmwareResponse. **3.** The Charging station sends a FirmwareStatusNotificationRequest with status *Downloading*. **4.** The CSMS responds with a FirmwareStatusNotificationResponse. **5.** The Charging station sends a FirmwareStatusNotificationRequest with status *Downloaded*. **6.** The CSMS responds with a FirmwareStatusNotificationResponse. **7.** The Charging station sends a FirmwareStatusNotificationRequest with status *Installing*. **8.** The CSMS responds with a FirmwareStatusNotificationResponse. **9.** The Charging station sends a FirmwareStatusNotificationRequest with status *Installed*. **10.** The CSMS responds with a FirmwareStatusNotificationResponse.
	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.
7	**Error handling**	n/a
8	**Remark(s)**	Measures SHOULD be taken to secure the firmware when it is stored on a server or workstation. When migrating to a new version of OCPP it is RECOMMENDED to install a fallback NetworkConnectionProfile with the new configuration. FTP needs to be able to use Passive FTP, to be able to transverse over as many different typologies as possible.

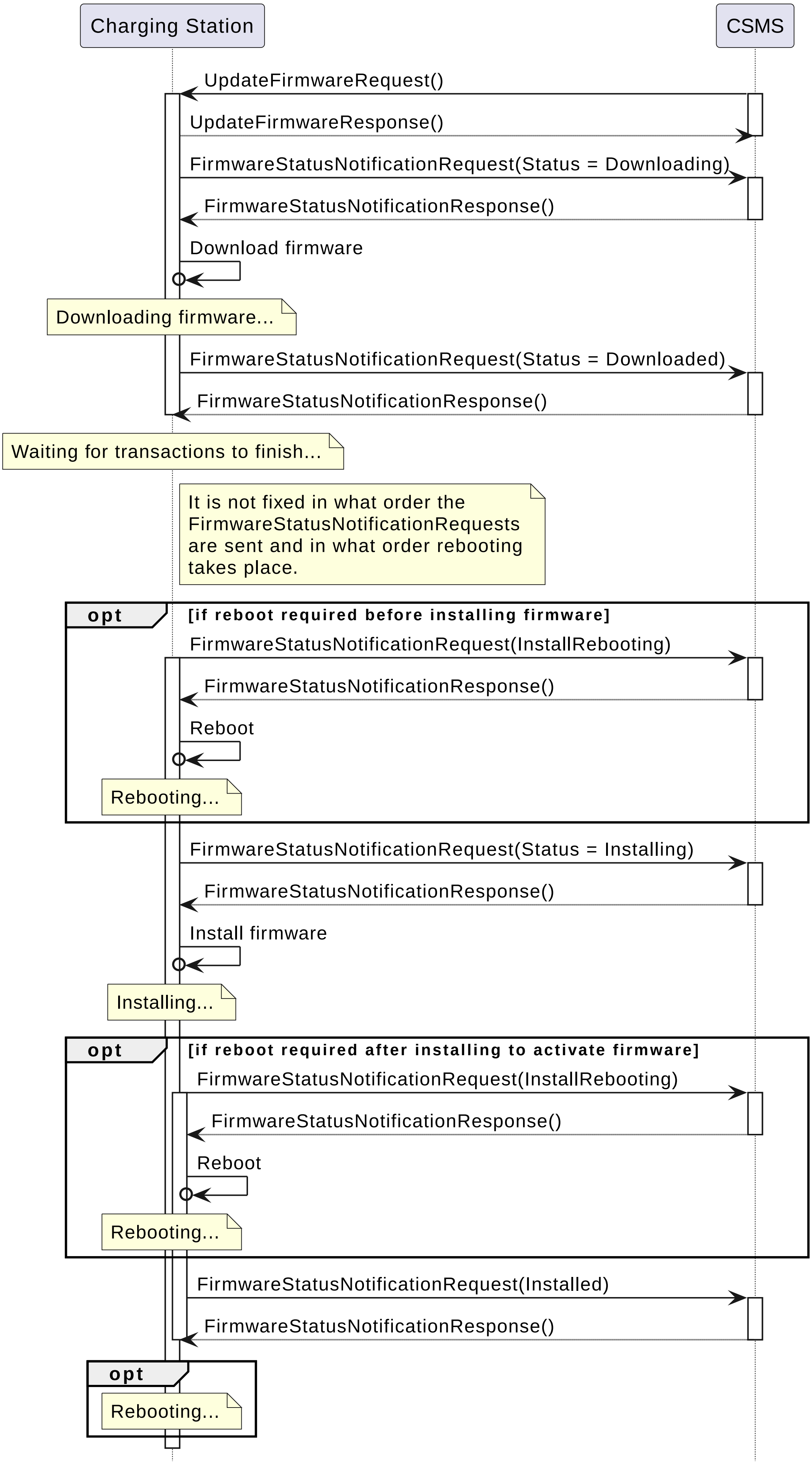

Charging Station
CSMS
UpdateFirmwareRequest()
UpdateFirmwareResponse()
FirmwareStatusNotificationRequest(Status = Downloading)
FirmwareStatusNotificationResponse()
Download firmware
Downloading firmware...
FirmwareStatusNotificationRequest(Status = Downloaded)
FirmwareStatusNotificationResponse()
Waiting for transactions to finish...
It is not fixed in what order the FirmwareStatusNotificationRequests are sent and in what order rebooting takes place.
opt
[if reboot required before installing firmware]
FirmwareStatusNotificationRequest(InstallRebooting)
FirmwareStatusNotificationResponse()
Reboot
Rebooting...
FirmwareStatusNotificationRequest(Status = Installing)
FirmwareStatusNotificationResponse()
Install firmware
Installing...
opt
[if reboot required after installing to activate firmware]
FirmwareStatusNotificationRequest(InstallRebooting)
FirmwareStatusNotificationResponse()
Reboot
Rebooting...
FirmwareStatusNotificationRequest(Installed)
FirmwareStatusNotificationResponse()
opt
Rebooting...

Figure 150. Sequence diagram Non-Secure firmware upgrade

L02 - Non-Secure Firmware Update - Requirements

Table 122. 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.	Same as L01.FR.01
L02.FR.02	When the Charging Station has successfully downloaded the new firmware AND (*installDateTime* is not set OR current time >= *installDateTime*)	The Charging Station SHALL install the new firmware as soon as it is able to.	Same as L01.FR.04 and L01.FR.05
L02.FR.03	L02.FR.02 AND The Charging Station has ongoing transactions AND When it is not possible to start installation of firmware while a transaction is ongoing	The Charging Station SHALL wait until all transactions have ended, before commencing installation.	Same as L01.FR.06
L02.FR.04	L02.FR.03 or L02.FR.22 AND configuration variable `AllowNewSessionsPendingFirmwareUpdate` is *false* or does not exist	The Charging Station SHALL set all EVSE that are not in use to UNAVAILABLE while the Charging Station waits for the ongoing transactions to end. Until the firmware is installed, any EVSE that becomes available SHALL be set to UNAVAILABLE.	Same as L01.FR.07
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.	Same as L01.FR.08
L02.FR.06		Every FirmwareStatusNotificationRequest sent for a firmware update SHALL contain the same requestId as the UpdateFirmwareRequest that started this firmware update.	Same as L01.FR.10
L02.FR.07	When the Charging Station does not start downloading firmware, because it is busy charging or because *retrieveDateTime* is in the future	The Charging Station SHALL send a FirmwareStatusNotificationRequest with status DownloadScheduled.	Same as L01.FR.13
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. Same as L01.FR.14
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.	Same as L01.FR.15
L02.FR.10	When the Charging Station has successfully downloaded the new firmware AND *installDateTime* 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.	Same as L01.FR.04 and L01.FR.16
L02.FR.14		The field *requestId* in FirmwareStatusNotificationRequest is mandatory, unless *status* = `Idle`.	Same as L01.FR.20

ID	Precondition	Requirement definition	Note
L02.FR.15	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. Same as L01.FR.24
L02.FR.16	Charging Station receives a TriggerMessageRequest for `FirmwareStatusNotification` AND last sent FirmwareStatusNotificationRequest had *status* = `Installed`	Charging Station SHALL return a FirmwareStatusNotificationRequest with *status* = `Idle`.	Same as L01.FR.25
L02.FR.17	Charging Station receives a TriggerMessageRequest for `FirmwareStatusNotification` AND last sent FirmwareStatusNotificationRequest had NOT *status* `Installed`	Charging Station SHALL return a FirmwareStatusNotificationRequest with the last sent *status*.	Same as L01.FR.26
L02.FR.18	L02.FR.15 AND the Charging Station is unable to cancel the firmware installation	The Charging Station MAY respond with *status* = `Rejected`.	Same as L01.FR.27
L02.FR.19	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. Same as L01.FR.30
L02.FR.20	When the Charging Station has successfully installed and activated the new firmware	The Charging Station SHALL send a FirmwareStatusNotificationRequest with status Installed.	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 either: (a) send an optional FirmwareStatusNotificationRequest with *status* = `InstallRebooting` before rebooting and send a mandatory FirmwareStatusNotificationRequest with *status* = `Installed` by the newly activated firmware, or (b) only send a FirmwareStatusNotificationRequest with status set to `Installed` without reporting the reboot and activation of the new firmware.	Option (a) is preferred, because it notifies CSMS of an upcoming reboot of the Charging Station, and the final *status* = `Installed` is sent by the new firmware image, so that CSMS can be sure that the new firmware is active. This is not guaranteed by option (b) when rebooting of the new firmware should fail.
L02.FR.22	L02.FR.02 AND The Charging Station has ongoing transactions AND a reboot is needed to activate the installed firmware	The Charging Station SHALL wait until all transactions have ended, before activating the installed firmware.	E.g. in case of A/B firmware updates.
L02.FR.23	When the Charging Station has successfully downloaded the firmware AND *installDateTime* is not set AND Charging Station is waiting for a transaction to finish	The Charging Station MAY send a FirmwareStatusNotificationRequest with status InstallScheduled.	The case where *installDateTime* is set is covered by L02.FR.10.

L03 - Publish Firmware file on Local Controller

No.	Type	Description
1	**Name**	Publish Firmware file on Local Controller.
2	**ID**	L03
3	**Objective(s)**	To allow Charging Stations to download a firmware update directly from the Local Controller.
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	**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)**	**Successful postcondition:** The firmware is successfully published by the Local Controller. **Failure postcondition:** The Local Controller could not download the firmware file, and has sent the *DownloadFailed* status. The Local Controller could not verify the MD5 checksum, and has sent the *InvalidChecksum* status. The Local Controller could not publish the firmware file, and has sent the *PublishFailed* status.
7	**Error handling**	n/a
8	**Remark(s)**	For information about MD5 checksum see RFC-1321 [RFC1321].

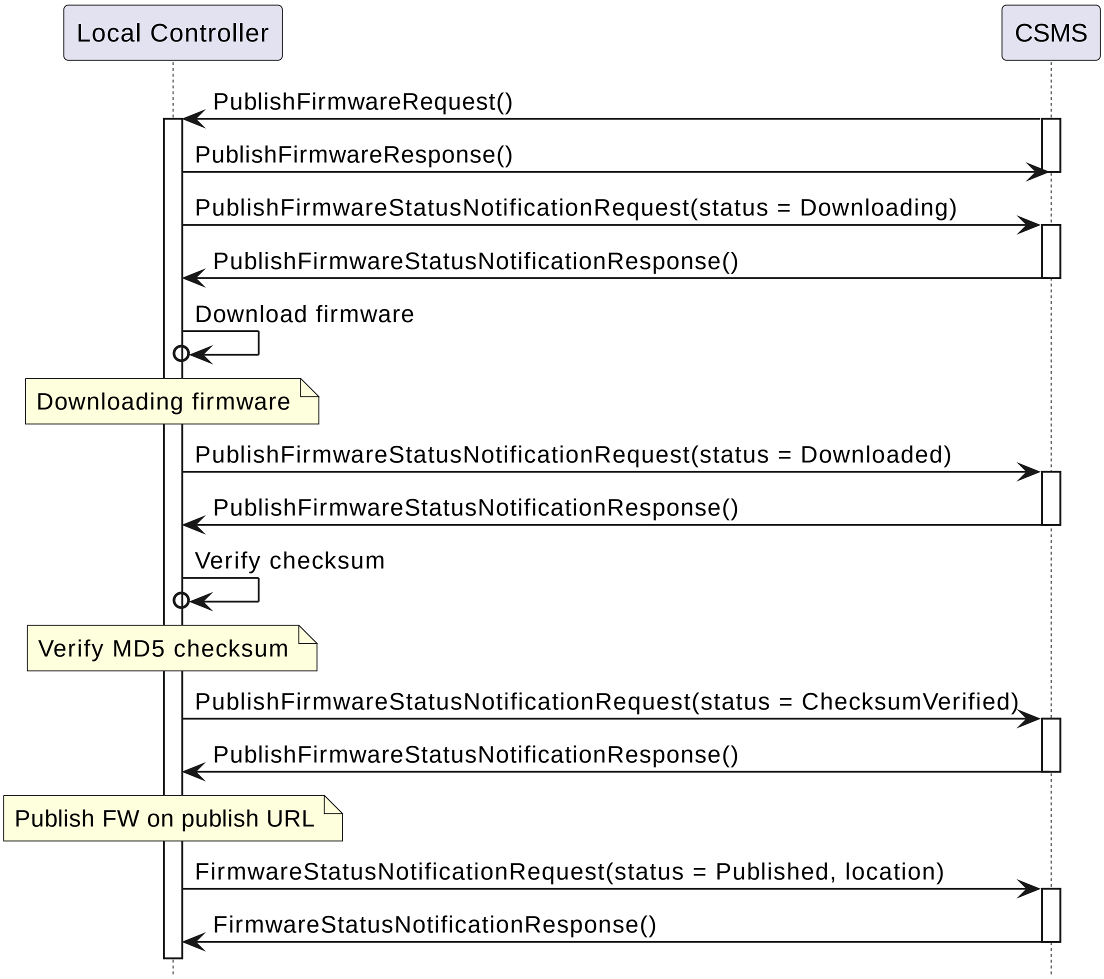

Local Controller
CSMS
PublishFirmwareRequest()
PublishFirmwareResponse()
PublishFirmwareStatusNotificationRequest(status = Downloading)
PublishFirmwareStatusNotificationResponse()
Download firmware
Downloading firmware
PublishFirmwareStatusNotificationRequest(status = Downloaded)
PublishFirmwareStatusNotificationResponse()
Verify checksum
Verify MD5 checksum
PublishFirmwareStatusNotificationRequest(status = ChecksumVerified)
PublishFirmwareStatusNotificationResponse()
Publish FW on publish URL
FirmwareStatusNotificationRequest(status = Published, location)
FirmwareStatusNotificationResponse()

Figure 151. Sequence Diagram: showing publishing of firmware (happy flow)

L03 - Publish Firmware file on Local Controller - Requirements

Table 123. 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 *location* field of the PublishFirmwareStatusNotificationRequest message with status *Published*.
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 *DownloadFailed*.
L03.FR.07	If the Local Controller cannot verify the MD5 checksum.	The Local Controller SHALL send a PublishFirmwareStatusNotificationRequest with status *InvalidChecksum*.
L03.FR.08	If the Local Controller cannot publish the firmware file.	The Local Controller SHALL send a PublishFirmwareStatusNotificationRequest with status *PublishFailed*.
L03.FR.09	After successfully publishing the firmware file.	The Local Controller SHALL send a PublishFirmwareStatusNotificationRequest with status *Published*.

ID	Precondition	Requirement definition
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* = `Idle`.
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

No.	Type	Description
1	**Name**	Unpublish Firmware file on Local Controller.
2	**ID**	L04
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	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.
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
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.

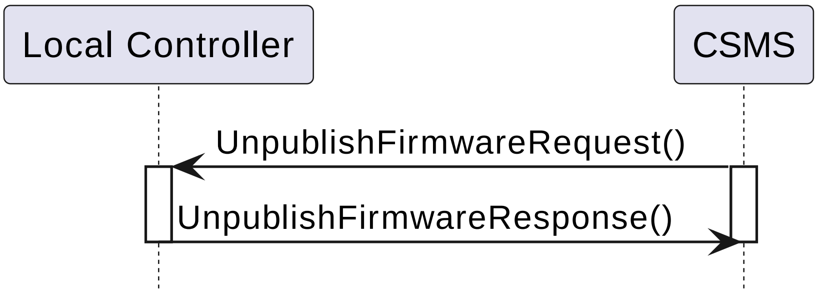


Figure 152. Sequence Diagram: Unpublishing a firmware file

L04 - Unpublish Firmware file on Local Controller - Requirements

Table 124. 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 *Unpublished*.
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 *NoFirmware*.
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 *Downloading* status AND **not** unpublish the firmware file.

M. Certificate Management

Chapter 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 bidirectional 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 figures: these sequences only apply for AC charging, although the certificate handling (which is the focus in this section) does not differ in AC or DC.

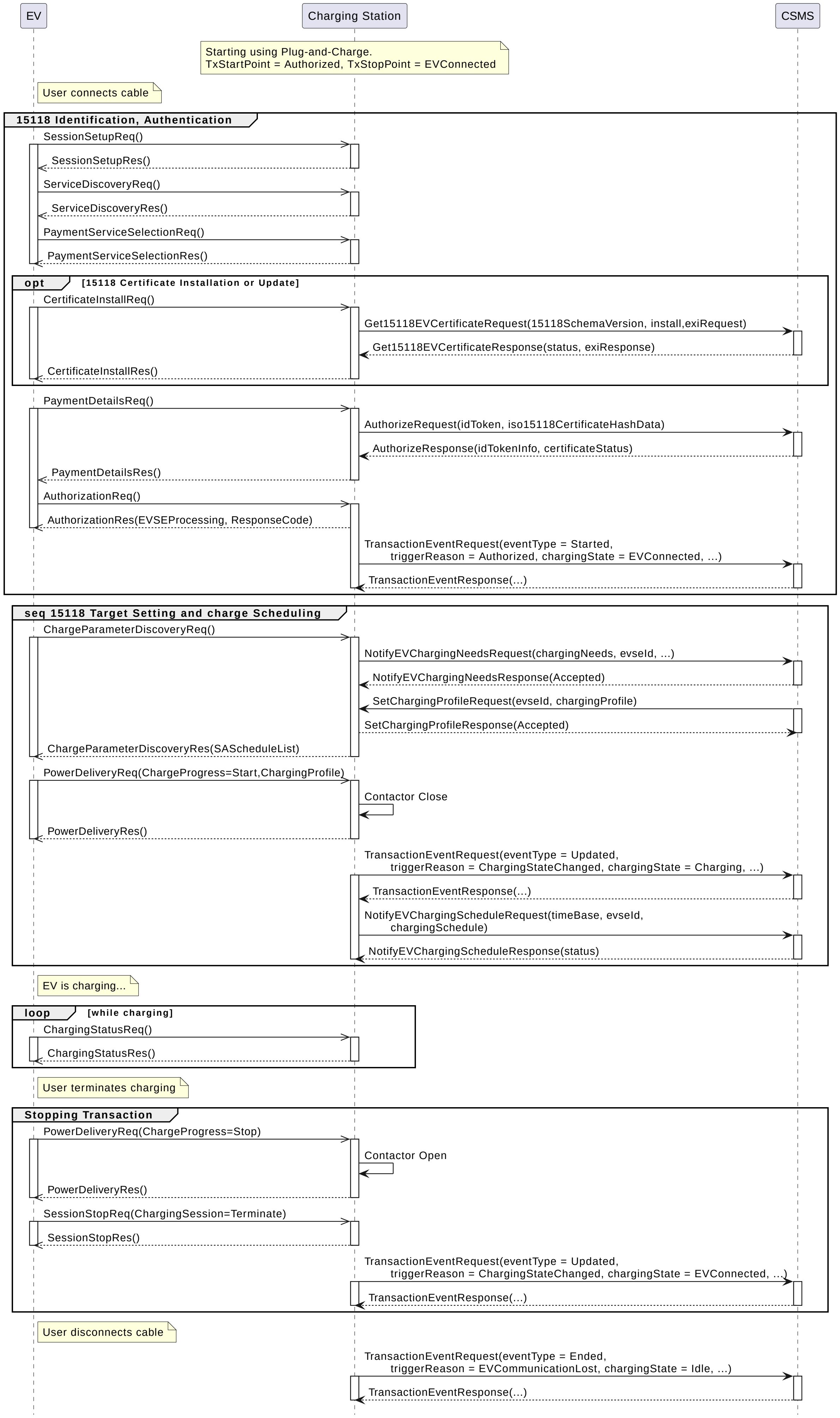

EV
Charging Station
CSMS
Starting using Plug-and-Charge.
TxStartPoint = Authorized, TxStopPoint = EVConnected
User connects cable
15118 Identification, Authentication
SessionSetupReq()
SessionSetupRes()
ServiceDiscoveryReq()
ServiceDiscoveryRes()
PaymentServiceSelectionReq()
PaymentServiceSelectionRes()
opt [15118 Certificate Installation or Update]
CertificateInstallReq()
Get15118EVCertificateRequest(15118SchemaVersion, install,exiRequest)
Get15118EVCertificateResponse(status, exiResponse)
CertificateInstallRes()
PaymentDetailsReq()
AuthorizeRequest(idToken, iso15118CertificateHashData)
AuthorizeResponse(idTokenInfo, certificateStatus)
PaymentDetailsRes()
AuthorizationReq()
AuthorizationRes(EVSEProcessing, ResponseCode)
TransactionEventRequest(eventType = Started,
triggerReason = Authorized, chargingState = EVConnected, ...)
TransactionEventResponse(...)
seq 15118 Target Setting and charge Scheduling
ChargeParameterDiscoveryReq()
NotifyEVChargingNeedsRequest(chargingNeeds, evseId, ...)
NotifyEVChargingNeedsResponse(Accepted)
SetChargingProfileRequest(evseId, chargingProfile)
SetChargingProfileResponse(Accepted)
ChargeParameterDiscoveryRes(SAScheduleList)
PowerDeliveryReq(ChargeProgress=Start,ChargingProfile)
Contactor Close
PowerDeliveryRes()
TransactionEventRequest(eventType = Updated,
triggerReason = ChargingStateChanged, chargingState = Charging, ...)
TransactionEventResponse(...)
NotifyEVChargingScheduleRequest(timeBase, evseId,
chargingSchedule)
NotifyEVChargingScheduleResponse(status)
EV is charging...
loop [while charging]
ChargingStatusReq()
ChargingStatusRes()
User terminates charging
Stopping Transaction
PowerDeliveryReq(ChargeProgress=Stop)
Contactor Open
PowerDeliveryRes()
SessionStopReq(ChargingSession=Terminate)
SessionStopRes()
TransactionEventRequest(eventType = Updated,
triggerReason = ChargingStateChanged, chargingState = EVConnected, ...)
TransactionEventResponse(...)
User disconnects cable
TransactionEventRequest(eventType = Ended,
triggerReason = EVCommunicationLost, chargingState = Idle, ...)
TransactionEventResponse(...)

Figure 153. Sequence with Authorization and Scheduling with ISO 15118-2

NOTE	The time-out on the ChargeParameterDiscoveryReq is 2 seconds, but this can be prolonged up to 60 seconds to wait for charging profile (SAScheduleList) to be provided by the CSMS. See ISO 15118-2 [ISO15118-2].

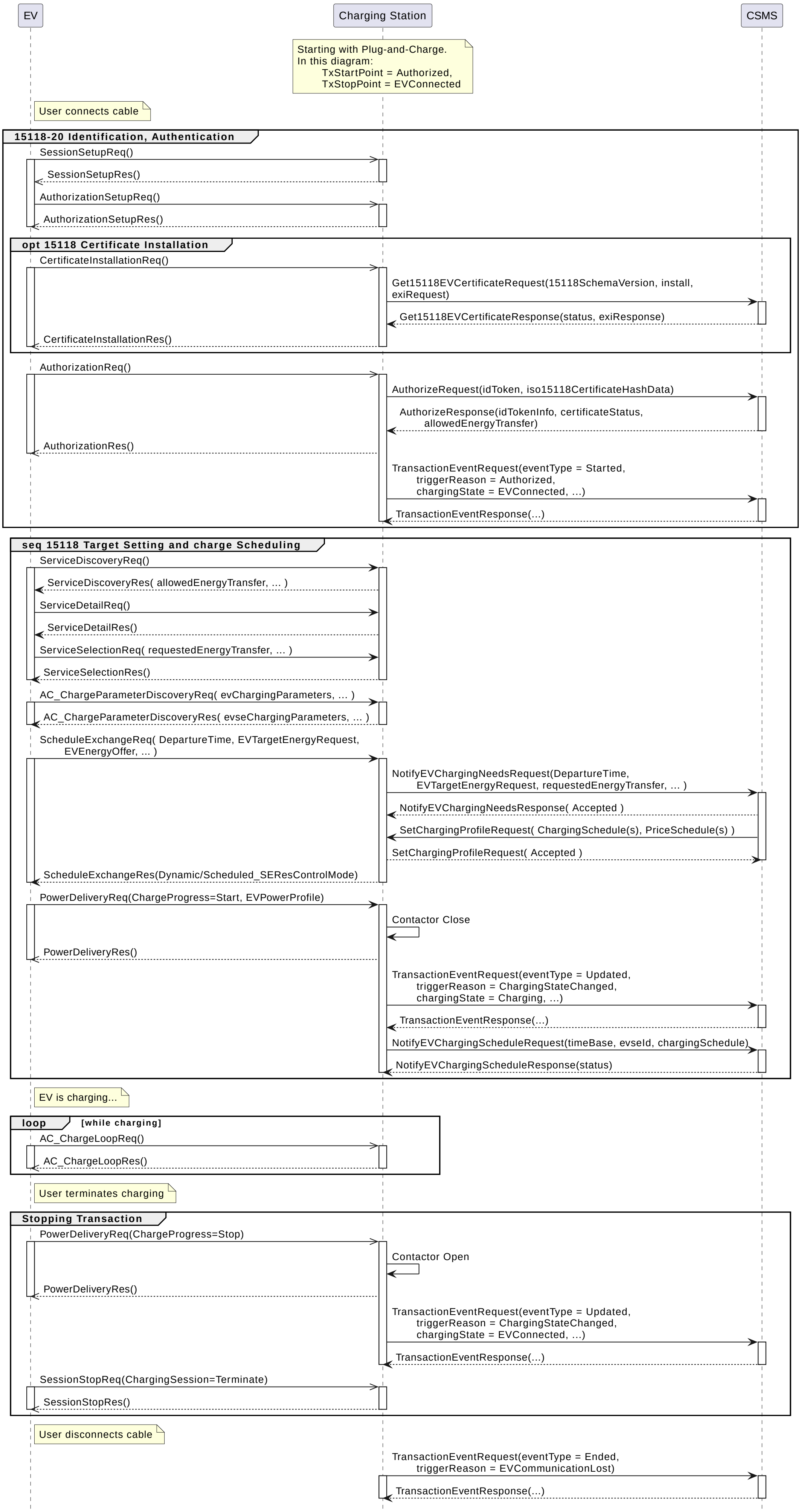

EV
Charging Station
CSMS
Starting with Plug-and-Charge.
In this diagram:
TxStartPoint = Authorized,
TxStopPoint = EVConnected
User connects cable
15118-20 Identification, Authentication
SessionSetupReq()
SessionSetupRes()
AuthorizationSetupReq()
AuthorizationSetupRes()
opt 15118 Certificate Installation
CertificateInstallationReq()
Get15118EVCertificateRequest(15118SchemaVersion, install, exiRequest)
Get15118EVCertificateResponse(status, exiResponse)
CertificateInstallationRes()
AuthorizationReq()
AuthorizeRequest(idToken, iso15118CertificateHashData)
AuthorizeResponse(idTokenInfo, certificateStatus, allowedEnergyTransfer)
AuthorizationRes()
TransactionEventRequest(eventType = Started, triggerReason = Authorized, chargingState = EVConnected, ...)
TransactionEventResponse(...)
seq 15118 Target Setting and charge Scheduling
ServiceDiscoveryReq()
ServiceDiscoveryRes(allowedEnergyTransfer, ...)
ServiceDetailReq()
ServiceDetailRes()
ServiceSelectionReq(requestedEnergyTransfer, ...)
ServiceSelectionRes()
AC_ChargeParameterDiscoveryReq(evChargingParameters, ...)
AC_ChargeParameterDiscoveryRes(evseChargingParameters, ...)
ScheduleExchangeReq(DepartureTime, EVTargetEnergyRequest, EVEnergyOffer, ...)
NotifyEVChargingNeedsRequest(DepartureTime, EVTargetEnergyRequest, requestedEnergyTransfer, ...)
NotifyEVChargingNeedsResponse(Accepted)
SetChargingProfileRequest(ChargingSchedule(s), PriceSchedule(s))
SetChargingProfileRequest(Accepted)
ScheduleExchangeRes(Dynamic/Scheduled_SEResControlMode)
PowerDeliveryReq(ChargeProgress=Start, EVPowerProfile)
Contactor Close
PowerDeliveryRes()
TransactionEventRequest(eventType = Updated, triggerReason = ChargingStateChanged, chargingState = Charging, ...)
TransactionEventResponse(...)
NotifyEVChargingScheduleRequest(timeBase, evseId, chargingSchedule)
NotifyEVChargingScheduleResponse(status)
EV is charging...
loop [while charging]
AC_ChargeLoopReq()
AC_ChargeLoopRes()
User terminates charging
Stopping Transaction
PowerDeliveryReq(ChargeProgress=Stop)
Contactor Open
PowerDeliveryRes()
TransactionEventRequest(eventType = Updated, triggerReason = ChargingStateChanged, chargingState = EVConnected, ...)
TransactionEventResponse(...)
SessionStopReq(ChargingSession=Terminate)
SessionStopRes()
User disconnects cable
TransactionEventRequest(eventType = Ended, triggerReason = EVCommunicationLost)
TransactionEventResponse(...)

Figure 154. Sequence for authorization and scheduling with ISO 15118-20

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.

Chapter 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).

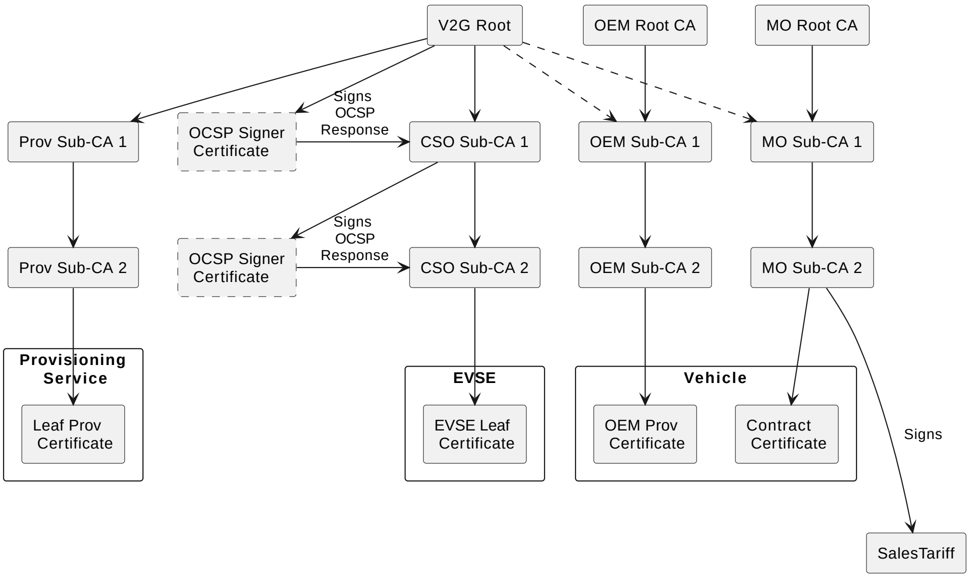


Figure 155. PKIs applied for Plug & Charge identification mode

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
V2GChargingStationCertificate	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.

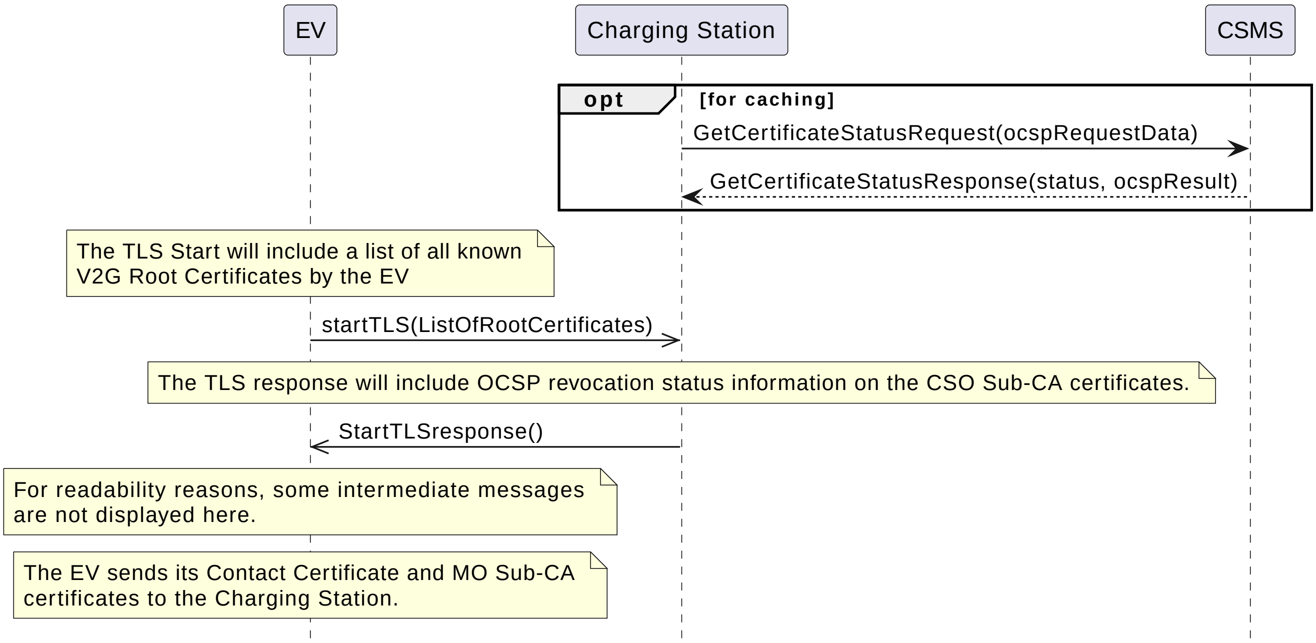

EV
Charging Station
CSMS
opt
[for caching]
GetCertificateStatusRequest(ocspRequestData)
GetCertificateStatusResponse(status, ocspResult)
The TLS Start will include a list of all known V2G Root Certificates by the EV
startTLS(ListOfRootCertificates)
The TLS response will include OCSP revocation status information on the CSO Sub-CA certificates.
StartTLSresponse()
For readability reasons, some intermediate messages are not displayed here.
The EV sends its Contact Certificate and MO Sub-CA certificates to the Charging Station.

Figure 156. 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 125. 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 *V2GChargingStationCertificate* and *V2GRootCertificate*. May be used during TLS setup between EV and Charging Station.

Chapter 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 126. 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 section 5.3. ISO 15118 based Smart Charging.

Chapter 4. Use cases & Requirements

M01 - Certificate installation EV

Updated in OCPP 2.1

No.	Type	Description
1	**Name**	Certificate Installation
2	**ID**	M01
	Reference	ISO15118-1 C2
3	**Objective(s)**	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 Get15118EVCertificateRequest message **with *action* = `Install`** to the CSMS. - The CSMS responds with Get15118EVCertificateResponse 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 CertificateInstallationRequest, for example a contract certificate pool.
6	**Postcondition(s)**	See ISO15118-1, use case End conditions C2, page 23.
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 CertificateInstallationReq is 5 seconds. There may be alternative communication paths for doing a certificate installation. However, these are outside the scope of this standard.

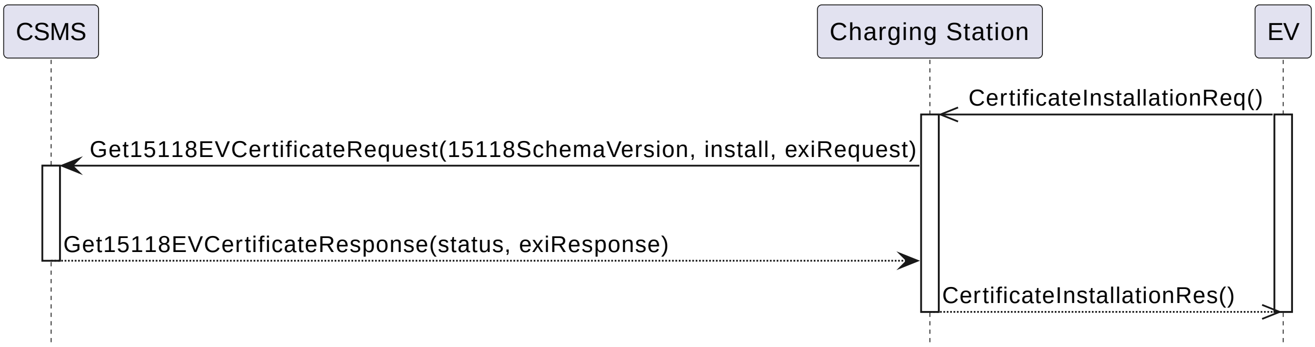

CSMS
Charging Station
EV
CertificateInstallationReq()
Get15118EVCertificateRequest(15118SchemaVersion, install, exiRequest)
Get15118EVCertificateResponse(status, exiResponse)
CertificateInstallationRes()

Figure 157. Certificate installation with ISO 15118-2

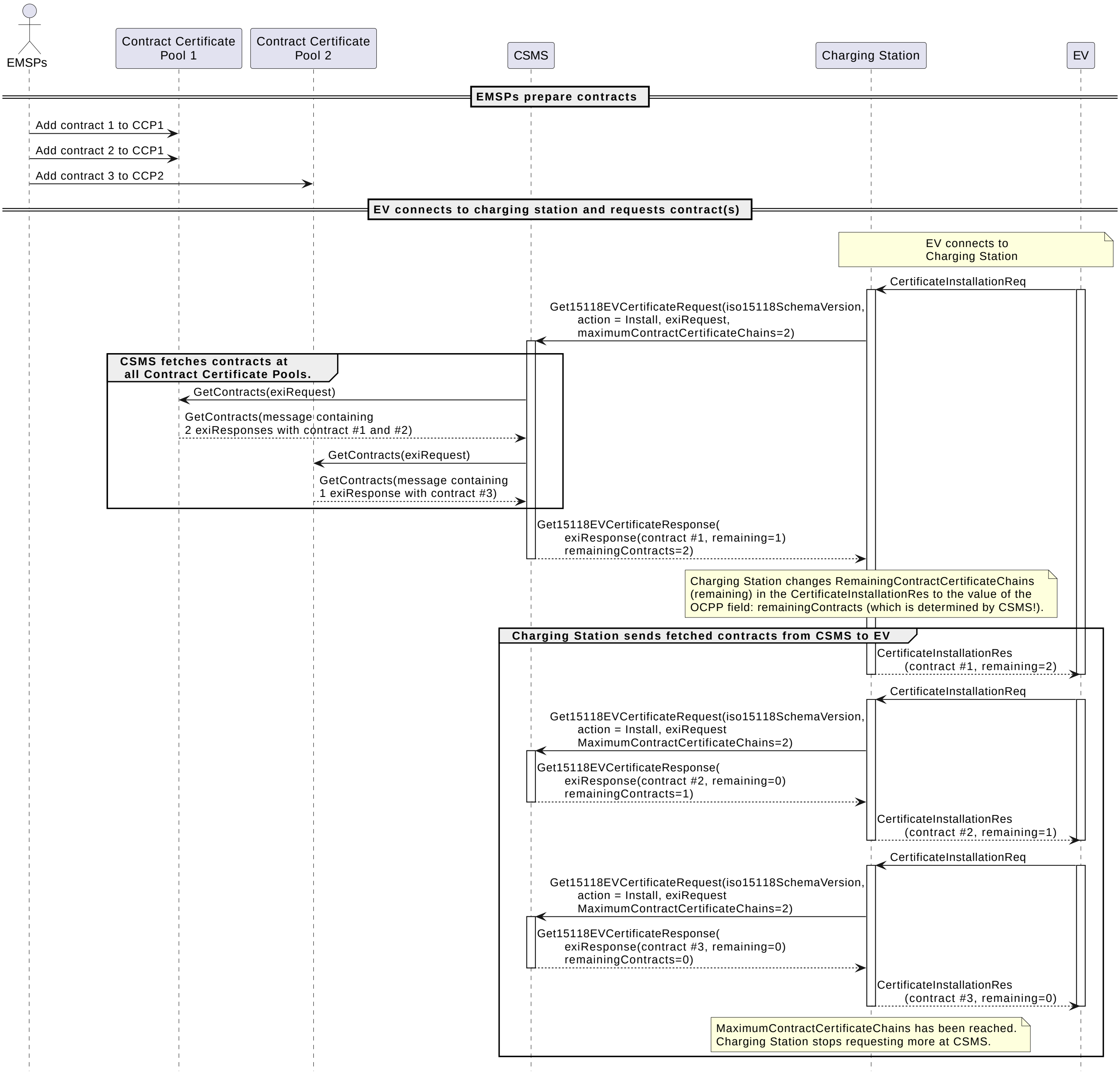

EMSPs
Contract Certificate Pool 1
Contract Certificate Pool 2
CSMS
Charging Station
EV
EMSPs prepare contracts
Add contract 1 to CCP1
Add contract 2 to CCP1
Add contract 3 to CCP2
EV connects to charging station and requests contract(s)
EV connects to Charging Station
CertificateInstallationReq
Get15118EVCertificateRequest(iso15118SchemaVersion, action = Install, exiRequest, maximumContractCertificateChains=2)
CSMS fetches contracts at all Contract Certificate Pools.
GetContracts(exiRequest)
GetContracts(message containing 2 exiResponses with contract #1 and #2)
GetContracts(exiRequest)
GetContracts(message containing 1 exiResponse with contract #3)
Get15118EVCertificateResponse(exiResponse(contract #1, remaining=1) remainingContracts=2)
Charging Station changes RemainingContractCertificateChains (remaining) in the CertificateInstallationRes to the value of the OCPP field: remainingContracts (which is determined by CSMS!).
Charging Station sends fetched contracts from CSMS to EV
CertificateInstallationRes (contract #1, remaining=2)
CertificateInstallationReq
Get15118EVCertificateRequest(iso15118SchemaVersion, action = Install, exiRequest MaximumContractCertificateChains=2)
Get15118EVCertificateResponse(exiResponse(contract #2, remaining=0) remainingContracts=1)
CertificateInstallationRes (contract #2, remaining=1)
CertificateInstallationReq
Get15118EVCertificateRequest(iso15118SchemaVersion, action = Install, exiRequest MaximumContractCertificateChains=2)
Get15118EVCertificateResponse(exiResponse(contract #3, remaining=0) remainingContracts=0)
CertificateInstallationRes (contract #3, remaining=0)
MaximumContractCertificateChains has been reached. Charging Station stops requesting more at CSMS.

Figure 158. Getting multiple contract certificates with ISO 15118-20

Source: ISO15118-1

M01 - Certificate installation - Requirements

Table 127. M01 - Requirements

ID	Precondition	Requirement definition	Note
M01.FR.01	Upon receiving an ISO 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.
M01.FR.02 **(2.1)**	When EV uses ISO 15118-2	Charging Station SHALL NOT include the field *maximumContractCertificateChains* in the Get15118EVCertificateRequest message.	

ID	Precondition	Requirement definition	Note
M01.FR.03 **(2.1)**	When EV uses ISO 15118-20	Charging Station SHALL set field *maximumContractCertificateChains* in the Get15118EVCertificateRequest to the value of MaximumContractCertificateChains in the ISO 15118-20 CertificateInstallationReq message.	
M01.FR.04 **(2.1)**	When *maximumContractCertificateChains* is present in Get15118EVCertificateRequest AND CSMS has received all CertificateInstallationRes responses from the contract certificate pool(s)	CSMS SHALL return the first CertificateInstallationRes response and the total number of available contracts in *remainingContracts* in Get15118EVCertificateResponse.	Presence of field *maximumContractCertificateChains* means that this is for ISO 15118-20.
M01.FR.05 **(2.1)**	When EV uses ISO 15118-20 AND Upon receiving a Get15118EVCertificateResponse from CSMS	Charging station SHALL update the value of RemainingContractCertificateChains in the ISO 15118-20 CertificateInstallationRes message to EV to the value of *remainingContracts* of the OCPP Get15118EVCertificateResponse message.	
M01.FR.06 **(2.1)**	When EV uses ISO 15118-20 AND As long as *remainingContracts* in Get15118EVCertificateResponse is larger than 0	Charging station SHALL send the same Get15118EVCertificateRequest to CSMS	CSMS will send the next CertificateInstallationRes in the Get15118EVCertificateResponse.
M01.FR.07 **(2.1)**	When EV uses ISO 15118-20 AND M01.FR.06	CSMS SHALL return the next CertificateInstallationRes and decrease the value of *remainingContracts* in Get15118EVCertificateResponse by 1.	

M02 - Certificate Update EV

No.	Type	Description
1	**Name**	Certificate Update
2	**ID**	M02
	Reference	ISO15118-1 C1
3	**Objective(s)**	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	**15118**: See ISO15118-1, use case Objective C1, Scenario Description, first 3 bullets, page 21. **OCPP**: - The Charging Station sends a Get15118EVCertificateRequest message **with *action* = `Update`** to the CSMS. - The CSMS responds with Get15118EVCertificateResponse to the Charging Station. **15118**: See ISO15118-1, use case Description C1, Scenario Description, last 2 bullets, page 21.

No.	Type	Description
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 CertificateInstallationRequest, for example a contract certificate pool.
6	**Postcondition(s)**	See ISO15118-1, use case Objective C1 and C2, page 20/22.
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.

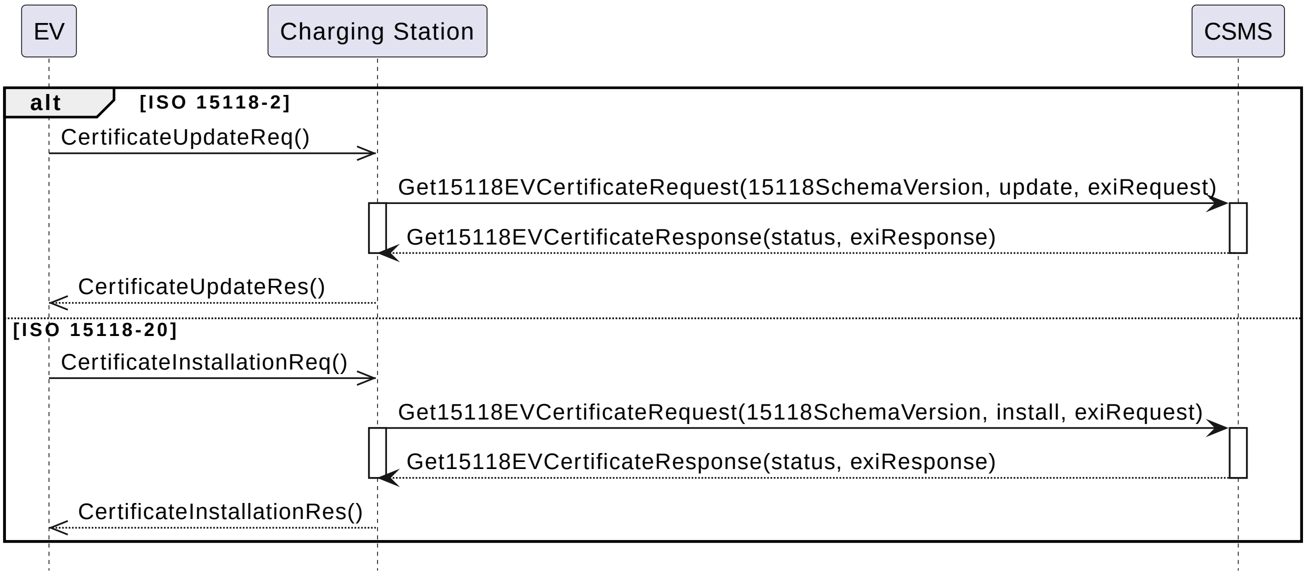

EV
Charging Station
CSMS
alt
[ISO 15118-2]
CertificateUpdateReq()
Get15118EVCertificateRequest(15118SchemaVersion, update, exiRequest)
Get15118EVCertificateResponse(status, exiResponse)
CertificateUpdateRes()
[ISO 15118-20]
CertificateInstallationReq()
Get15118EVCertificateRequest(15118SchemaVersion, install, exiRequest)
Get15118EVCertificateResponse(status, exiResponse)
CertificateInstallationRes()

Figure 159. Certificate Update

Source: ISO15118-1

M02 - Certificate Update - Requirements

Table 128. M02 - Requirements

ID	Precondition	Requirement definition	Note
M02.FR.01 **(2.1)**	Upon receiving an ISO 15118 CertificateUpdateReq AND EV uses ISO 15118-2	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.
M02.FR.02 **(2.1)**	Upon receiving an ISO 15118 CertificateUpdateReq AND EV uses ISO 15118-20	Charging Station SHALL act according to use case M01	ISO 15118-20 does not differentiate between Install and Update anymore.

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
3	**Objective(s)**	To enable the CSMS to retrieve a list of available certificates from a Charging Station.

No.	Type	Description
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	**1.** The CSMS requests the Charging Station to send a list of installed certificates by sending a GetInstalledCertificateIdsRequest **2.** The Charging Station responds with a GetInstalledCertificateIdsResponse
5	**Prerequisite(s)**	n/a
6	**Postcondition(s)**	The CSMS received a list of installed certificates
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.

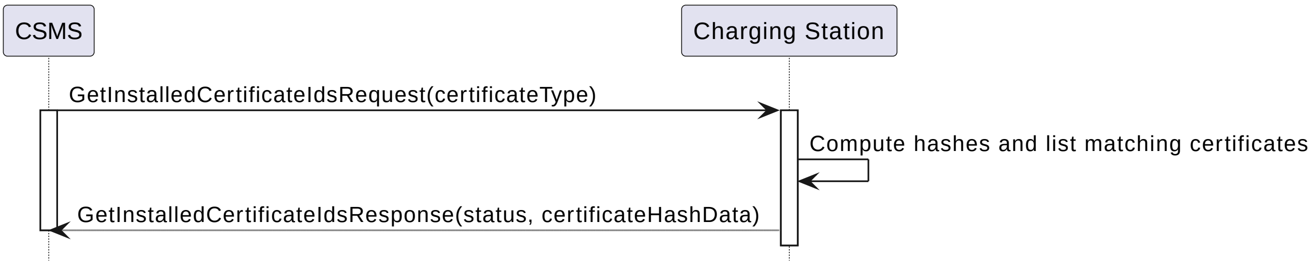

CSMS
Charging Station
GetInstalledCertificateIdsRequest(certificateType)
Compute hashes and list matching certificates
GetInstalledCertificateIdsResponse(status, certificateHashData)

Figure 160. Retrieve list of available certificates from a Charging Station

M03 - Retrieve list of available certificates from a Charging Station - Requirements

Table 129. M03 - Requirements

ID	Precondition	Requirement definition
M03.FR.01	After receiving a GetInstalledCertificateIdsRequest	The Charging Station SHALL respond with a GetInstalledCertificateIdsResponse.
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 *NotFound*.
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 *Accepted*.
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

No.	Type	Description
1	**Name**	Delete a specific certificate from a Charging Station
2	**ID**	M04
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

No.	Type	Description
	Scenario description	**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.
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. 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. 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.

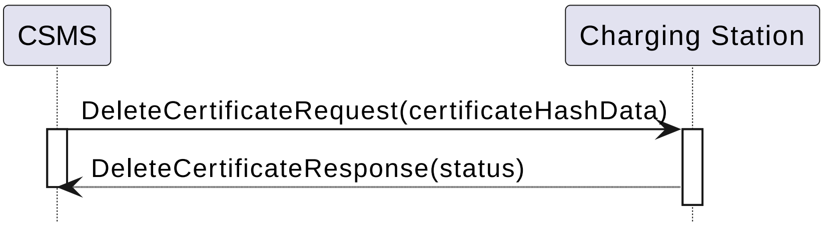

CSMS
Charging Station
DeleteCertificateRequest(certificateHashData)
DeleteCertificateResponse(status)

Figure 161. Delete Installed Certificate

M04 - Delete a specific certificate from a Charging Station - Requirements

Table 130. 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 *status* to `Accepted` 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 *status* to `Failed` 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 *certificateHashData* refers to the *Charging Station Certificate* (see use case A)	Charging Station SHALL respond with DeleteCertificateReponse with *status* = `Failed`.	Deletion of the *Charging Station Certificate* is not allowed via DeleteCertificateRequest.
M04.FR.07	When deleting a certificate	The CSMS SHALL use the same *hashAlgorithm* as the Charging Station uses to report the certificateHashData for the certificate in the GetInstalledCertificateIdsResponse.	This ensures CSMS uses a *hashAlgorithm* that is supported by the Charging Station.

ID	Precondition	Requirement definition	Note
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 orhan certificates that can no longer be deleted.

M05 - Install CA certificate in a Charging Station

No.	Type	Description
1	**Name**	Install CA certificate in a Charging Station
2	**ID**	M05
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	**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.
7	**Error handling**	n/a
8	**Remark(s)**	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. 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. It is allowed to have multiple certificates of the same type installed.

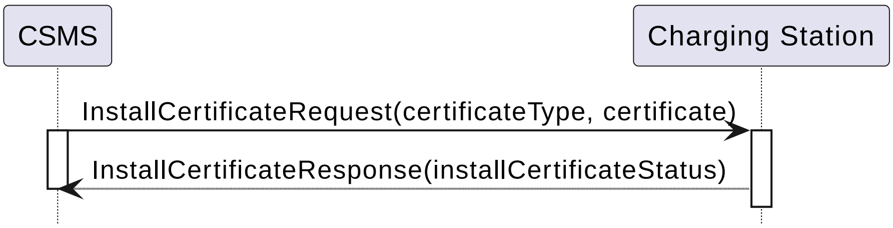

CSMS
Charging Station
InstallCertificateRequest(certificateType, certificate)
InstallCertificateResponse(installCertificateStatus)

Figure 162. Install CA certificate in a Charging Station

M05 - Install CA certificate in a Charging Station - Requirements

Table 131. 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 setting *status* to `Failed` in the InstallCertificateResponse.

ID	Precondition	Requirement definition
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* = `Rejected`.
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* = `Rejected`.
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* = `Accepted`
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.
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

No.	Type	Description
1	**Name**	Get V2G Charging Station Certificate status
2	**ID**	M06
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..

No.	Type	Description
	Actors	Charging Station, CSMS
	Scenario description	**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)**	**Successful postcondition:** The Charging Station received the OCSP certificate status for the requested certificate **Failure postcondition:** The retrieval of the OCSP certificate status by the CSMS failed
7	**Error handling**	n/a
8	**Remark(s)**	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. 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. 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. *responderURL* is required in OCPP, while it is optional in ISO 15118. Without a *responderURL* in a certificate it cannot work, so a *responderURL* is required for any certificate for which a GetCertificateStatusRequest can be expected.

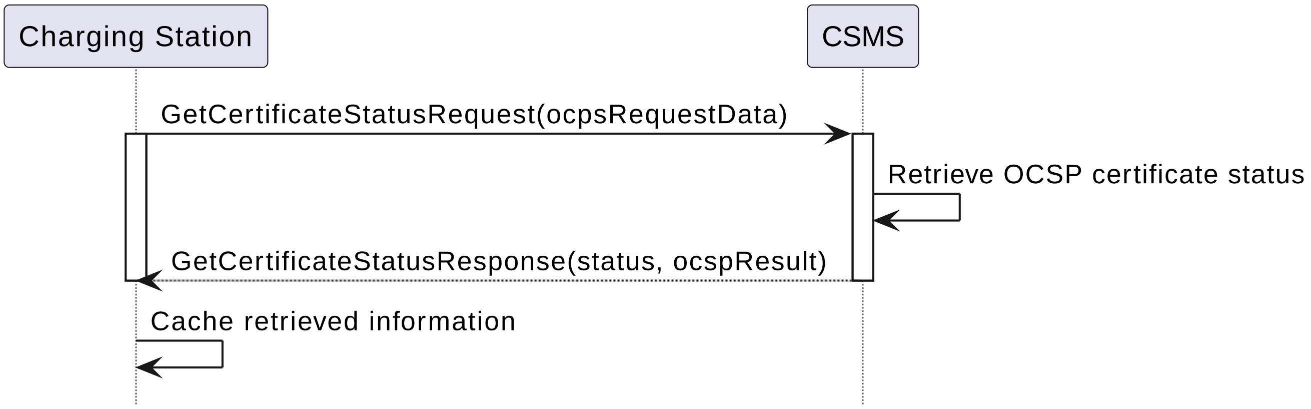

Charging Station
CSMS
GetCertificateStatusRequest(ocpsRequestData)
Retrieve OCSP certificate status
GetCertificateStatusResponse(status, ocspResult)
Cache retrieved information

Figure 163. Get V2G Charging Station Certificate status

M06 - Get V2G Charging Station Certificate status - Requirements

Table 132. 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 *status* to `Failed` in the GetCertificateStatusResponse.

ID	Precondition	Requirement definition
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.

M07 - Get Vehicle Certificate Chain Revocation Status

New in OCPP 2.1

No.	Type	Description
1	**Name**	Get Vehicle Certificate Chain Revocation Status
2	**ID**	M07
3	**Objective(s)**	To let Charging Station check validity of the vehicle certificate chain using OCSP and certificate revocation lists.
4	**Description**	Charging Station requests CSMS to check whether certificates are revoked. Depending on the information in the certificate, this requires checking via OCSP ("Authority Info Access" field in certificate) or CRL ("CRL Distribution Points" field). This check is required by ISO 15118-20 during the TLS handshake. The results are allowed to be cached at CSMS or Charging Station for at most one week by ISO 15118-20.
	Actors	Charging Station, CSMS
	Scenario description	**1.** Charging Station sends a GetCertificateChainStatusRequest with a list of certificates that need to be validated via an OCSP request and/or via their certificate revocation list. **2** For each certificate, CSMS first checks the status in cached data from earlier OCSP or CRL checks, less than a week old. **3** If the certificate is not present in the cached status information, CSMS will issue an OCSP request or download the CRL. **4** CSMS responds with a GetCertificateChainStatusResponse with the result for each certificate.
5	**Prerequisite(s)**	n/a
6	**Postcondition(s)**	When the request is successful the response contains the status for each certificate.
7	**Error handling**	If (part of) the request is not successful, the *status* of the certificates for which no status could be retrieved is `Failed`.
8	**Remark(s)**	Charging Stations that have unrestricted access to the internet might be able to fetch a CRL from a CRL distribution point themselves.

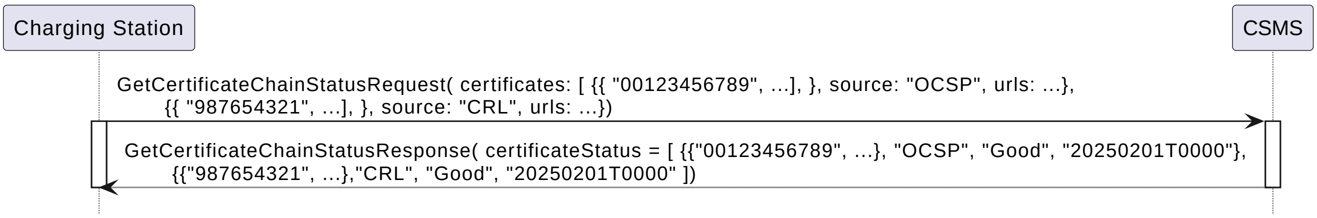

Charging Station
CSMS
GetCertificateChainStatusRequest(certificates: [{{ "00123456789", ...], }, source: "OCSP", urls: ...},
{{ "987654321", ...], }, source: "CRL", urls: ...})
GetCertificateChainStatusResponse(certificateStatus = [{{"00123456789", ...}, "OCSP", "Good", "20250201T0000"},
{{"987654321", ...},"CRL", "Good", "20250201T0000"])

Figure 164. Request Certificate Revocation List from CSMS

Request for two certificates

```
GetCertificateChainStatusRequest
{
    "certificateStatusRequests": [
    {
        "certificateHashData": {
            "hashAlgorithm": "SHA256"
            "issuerNameHash": "12345ABCDE" ,
            "issuerKeyHash": "ABCDE1234",
            "serialNumber": "00123456789"
        },
        "source": "OCSP",
        "urls": [
            "https://ocsp.responder.org/revoked.ocsp"
        ]
    }, {
        "certificateHashData": {
            "hashAlgorithm": "SHA256"
            "issuerNameHash": "ABCDE12345" ,
            "issuerKeyHash": "1234ABCDE",
            "serialNumber": "987634294239"
        },
        "source": "CRL",
        "urls": [
            "https://crls.com/revoked.crl",
            "ftp://ftp.crls.com/revoked.crl"
        ]
    } ]
}
```

Response

```
GetCertificateChainStatusResponse
{
    "certificateStatus": [
    {
        "certificateHashData": {
            "hashAlgorithm": "SHA256"
            "issuerNameHash": "12345ABCDE" ,
            "issuerKeyHash": "ABCDE1234",
            "serialNumber": "00123456789"
        },
        "source": "OCSP",
        "status": "Good",
        "nextUpdate": "20250201T1200Z"
    }, {
        "certificateHashData": {
            "hashAlgorithm": "SHA256"
            "issuerNameHash": "ABCDE12345" ,
            "issuerKeyHash": "1234ABCDE",
            "serialNumber": "987634294239"
        },
        "source": "CRL",
        "status": "Good",
        "nextUpdate": "20250201T1200Z"
    } ]
}
```

M07 - Get Vehicle Certificate Chain Revocation Status - Requirements

Table 133. M07 - Requirements

ID	Precondition	Requirement definition
M07.FR.01	When CSMS receives a GetCertificateChainStatusRequest with one or more entries in *certificateStatusRequests* which have *source* = `OCSP`	CSMS SHALL perform an OCSP request to one of URLs in *urls* for *certificateHashData* of each certificate.
M07.FR.02	M07.FR.01 AND the OCSP request to the selected URL from *urls* for *certificateHashData* fails	CSMS SHALL use *status* = `Failed` for this certificate with *certificateHashData* in the response.
M07.FR.03	M07.FR.01 AND NOT M07.FR.02	CSMS SHALL use the OCSP status (`Good/Revoked/Unknown`) for *status* of the certificate with *certificateHashData* in the response.
M07.FR.04	When CSMS receives a GetCertificateChainStatusRequest with one or more entries in *certificateStatusRequests* which have *source* = `CRL`	CSMS SHALL retrieve the CRL from one of the *urls* for *certificateHashData* of each certificate.
M07.FR.05	M07.FR.04 AND CSMS fails to download the CRL	CSMS SHALL use *status* = `Failed` for this certificate with *certificateHashData* in the response.
M07.FR.06	M07.FR.04 the *serialNumber* of a certificate is part of the downloaded CRL	CSMS SHALL use *status* = `Revoked` for this certificate in the response.
M07.FR.07	M07.FR.04 AND the *serialNumber* of a certificate is not part of the downloaded CRL	CSMS SHALL use *status* = `Good` for this certificate in the response.
M07.FR.08	When CSMS has checked all certificates in the GetCertificateChainStatusRequest	CSMS SHALL respond with a GetCertificateChainStatusResponse with a list of *certificateStatus* in which each entry has the *certificateHashData* of the certificate, the *source* (OCSP/CRL), *status* and *nextUpdate* of the checked certificates.
M07.FR.09	M07.FR.08	Charging Station SHOULD store the *certificateStatus* data from the GetCertificateChainStatusResponse in a local certificate status cache for at most one week.
M07.FR.10	When an entry for *certificateHashData* is present in the local certificate status cache AND the time in *nextUpdate* has not expired	Charging Station SHOULD NOT send a GetCertificateChainStatusRequest for the certificate.
M07.FR.11	When Charging Station wants to check the revocation status of a certificate chain AND the *certificateHashData* for one or more certificates of the chain is not present in the local certificate status cache	Charging Station SHALL send a GetCertificateChainStatusRequest to CSMS for the *certificateHashData* entries of the certificate chain that are not in the local certificate status cache.

N. Diagnostics

Chapter 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).

Chapter 2. Use cases & Requirements

2.1. Logging

N01 - Retrieve Log Information

Updated in OCPP 2.1

No.	Type	Description
1	**Name**	Retrieve Log
2	**ID**	N01
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	**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 `Uploading` **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 `Uploaded`. **7.** The CSMS responds with LogStatusNotificationResponse, acknowledging the status update request.
	Alternative scenario	*Requesting DataCollectorLog* The DataCollector is an optional component that collects samples of measurands at a high frequency, typically with subsecond intervals. It is a component that is reported as part of an EVSE and when enabled it only applies to that EVSE. The samples are logged internally in the Charging Station and can be retrieved in bulk with the GetLogRequest by specifying LogEnumType (changed) DataCollectorLog. For small intervals and lots of data it is much more efficient to use the DataCollector than installing a sampling monitor. * The DataCollectorLog is configured via device model component DataCollector. * Data collecting is switched on/off via DataCollector.Enabled. * Data collecting starts from DataCollector.DateTime["Start"] until DataCollector.DateTime["End"]. * Measurands defined in DataCollector.SampledMeasurands are collected at rate defined in DataCollector.SamplingInterval. Retrieval of the data collector log is done via a GetLogRequest with *logType* = `DataCollectorLog` following the steps described in *Scenario description* above.
5	**Prerequisite(s)**	- Diagnostics information is available for upload. - URL to upload file to is reachable and exists.
6	**Postcondition(s)**	**Successful postcondition:** Log file successfully uploaded. **Failure postcondition:** Log file not successfully uploaded and failed.
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.

No.	Type	Description
8	**Remark(s)**	As an example in this use case the requestId = 123, but this could be any value. 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. The FTP URL is of format: *ftp://User:password@host:port/path* in which the parts *User:password@*, *:password* or *:port* may be excluded. The Charging Station has a required Configuration Variable that reports which file transfer protocols it supports: `FileTransferProtocols` The format of the log file is not prescribed. FTP needs to be able to use Passive FTP, to be able to transverse over as much different typologies as possible.

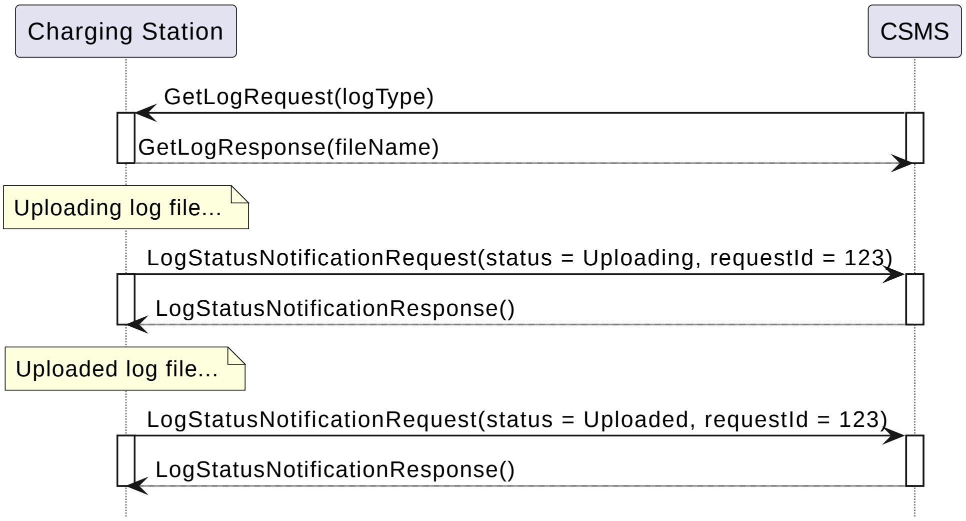

Charging Station
CSMS
GetLogRequest(logType)
GetLogResponse(fileName)
Uploading log file...
LogStatusNotificationRequest(status = Uploading, requestId = 123)
LogStatusNotificationResponse()
Uploaded log file...
LogStatusNotificationRequest(status = Uploaded, requestId = 123)
LogStatusNotificationResponse()

Figure 165. Sequence Diagram: Get Diagnostics

N01 - Retrieve Log Information - Requirements

Table 134. 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* `SecurityLog`	The Charging Station SHALL upload its security log	
N01.FR.04	N01.FR.02 AND The GetLogRequest contained *logType* `DiagnosticsLog`	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.	

ID	Precondition	Requirement definition	Note
N01.FR.08	When uploading a log document is started	The Charging Station SHALL send a LogStatusNotificationRequest with status `Uploading`.	
N01.FR.09	When a log document is uploaded successfully	The Charging Station SHALL send a LogStatusNotificationRequest with status `Uploaded`.	
N01.FR.10	When uploading a log document failed	The Charging Station SHALL send a LogStatusNotificationRequest with status `UploadFailure`, `BadMessage`, `PermissionDenied` OR `NotSupportedOperation`.	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.
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 `AcceptedCanceled`.	
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, then 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.

ID	Precondition	Requirement definition	Note
N01.FR.21	N01.FR.18 AND the GetLogRequest *log.remoteLocation* string ends with a slash character ("/")	The Charging Station SHALL append the filename that it returned in the GetLogResponse filename field to this string.	
N01.FR.22	N01.FR.18 AND the file to upload exceeds the threshold of 1MB (1024 * 1024 bytes)	The Charging Station MAY include the "Expect: 100-continue" header in the HTTP PUT request.	This gives the server a chance to deny the request before a lot of data is sent.
N01.FR.23	N01.FR.18	The Charging Station MAY implement a mechanism to resume HTTP uploads, if it can identify whether the upload server supports the same mechanism.	At the time of this writing, no official HTTP resume mechanism existed, only vendor-specific solutions and an IETF draft, which were not aligned.
N01.FR.24 **(2.1)**	N01.FR.02 AND The GetLogRequest contained *logType* `DataCollectorLog` AND The Charging Station supports the DataCollector log functionality.	The Charging Station SHALL upload its data collector log.	See DataCollector component in device model.
N01.FR.25 **(2.1)**	N01.FR.17 AND the GetLogRequest *log.remoteLocation* URL contains a userinfo component	The Charging Station SHALL use the given userinfo for HTTP Basic Authentication.	
N01.FR.26 **(2.1)**	N01.FR.18	The CSMS SHALL NOT send a log upload URL that points to a server responding with a redirection.	
N01.FR.27 **(2.1)**	N01.FR.18	The CSMS SHALL NOT send a log upload URL containing a fragment component.	The fragment component would not be part of the HTTP request, so it serves no purpose.
N01.FR.28 **(2.1)**	N01.FR.18 AND the CSMS includes a query component in the log upload URL	The CSMS SHALL NOT include a query component that ends with a slash character ("/").	This is to simplify processing of the URL string in the Charging Station.
N01.FR.29 **(2.1)**	N01.FR.18 AND the GetLogRequest *log.remoteLocation* string does not end with a slash character ("/")	The Charging Station SHALL use it as received without modifications.	
N01.FR.30 **(2.1)**	N01.FR.18	The Charging Station SHALL NOT follow redirects received in response to the request AND SHALL send a LogStatusNotificationRequest with *status* = `UploadFailure` and optionally add *reasonCode* = "RedirectNotAllowed".	Redirects bear the risk of changing the HTTP method and of redirection loops. This results in an upload failure as in N01.FR.10.

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 2.1: Part 1 - Architecture & Topology", chapter 4.

N02 - Get Monitoring report

Updated in OCPP 2.1

No.	Type	Description
1	**Name**	Get Monitoring Report
2	**ID**	N02
3	**Objective(s)**	To give the CSMS the ability to retrieve a report about configured monitoring settings per component and variable.

No.	Type	Description
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.
	Actors	Charging Station, CSMS, CSO
	Scenario description	**1.** The CSO triggers the CSMS to request a monitoring report from a Charging Station. **2.** The CSMS sends a GetMonitoringReportRequest to the Charging Station. **3.** The Charging Station responds with a GetMonitoringReportResponse. **4.** The Charging Station sends a NotifyMonitoringReportRequest to the CSMS. **5.** The CSMS responds with a NotifyMonitoringReportResponse. **6.** Steps #4 and #5 are repeated until all data of the monitoring report has been sent.
5	**Prerequisite(s)**	Charging Station supports Monitoring
6	**Postcondition(s)**	The CSMS received a report about the configured monitoring settings.
7	**Error handling**	n/a
8	**Remark(s)**	n/a

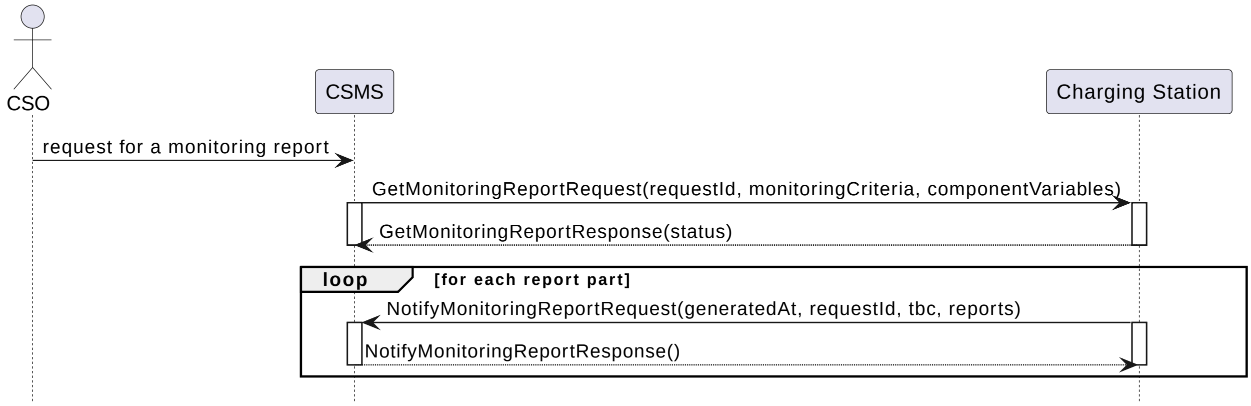

CSO
CSMS
Charging Station
request for a monitoring report
GetMonitoringReportRequest(requestId, monitoringCriteria, componentVariables)
GetMonitoringReportResponse(status)
loop
[for each report part]
NotifyMonitoringReportRequest(generatedAt, requestId, tbc, reports)
NotifyMonitoringReportResponse()

Figure 166. Sequence Diagram: Get Monitoring Report

N02 - Get Monitoring Report - Requirements

Table 135. N02 - Requirements

ID	Precondition	Requirement definition
N02.FR.01	NOT N02.FR.10 AND When the Charging Station receives a GetMonitoringReportRequest for supported *monitoringCriteria* OR without *monitoringCriteria*	The Charging Station SHALL send a getMonitoringReportResponse with Accepted.
N02.FR.02	When the Charging Station receives a GetMonitoringReportRequest for not supported *monitoringCriteria*	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 *requestId*	Every notifyMonitoringReportRequest sent for this GetMonitoringReportRequest SHALL contain the same *requestId*.
N02.FR.05	N02.FR.01 AND *monitoringCriteria* and *componentVariables* are NOT both empty.	The set of monitors reported in one or more notifyMonitoringReportRequest messages is limited to the set defined by *monitoringCriteria* and *componentVariables*.
N02.FR.06	N02.FR.01 AND *monitoringCriteria* is NOT empty AND *componentVariables* is empty.	The set of monitors reported in one or more notifyMonitoringReportRequest messages is limited to the set defined by *monitoringCriteria*.
N02.FR.07		The maximum number of *componentVariables* in one GetMonitoringReportRequest message is given by the `ItemsPerMessageGetReport` Configuration Variable

ID	Precondition	Requirement definition
N02.FR.08	N02.FR.01 AND *monitoringCriteria* is absent AND *componentVariables* is NOT empty.	The set of monitors reported in one or more notifyMonitoringReportRequest messages is limited to the set defined by *componentVariables*.
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 *monitoringCriteria* is empty AND *componentVariables* is empty.	The set of all existing monitors is reported in one or more notifyMonitoringReportRequest messages.
N02.FR.12	If *monitoringCriteria* contains `ThresholdMonitoring`	All monitors with *type* = `UpperThreshold` or *type* = `LowerThreshold` are reported.
N02.FR.13	If *monitoringCriteria* contains `DeltaMonitoring`	All monitors with *type* = `Delta` are reported.
N02.FR.14	If *monitoringCriteria* contains `PeriodicMonitoring`	All monitors with *type* = `Periodic` or *type* = `PeriodicClockAligned` are reported.
N02.FR.16	When Charging Station receives a GetMonitoringReportRequest with *componentVariable* elements in which *variable* is missing	The Charging Station SHALL report for every *variable* of the *component* in *componentVariable*.
N02.FR.17	When Charging Station receives a GetMonitoringReportRequest with *componentVariable* elements in which *variable* is present, but *instance* is missing	The Charging Station SHALL report for every instance of the *variable* of the *component* in *componentVariable*.
N02.FR.18	N02.FR.11 AND When Charging Station receives a GetMonitoringReportRequest with a *component* in a *componentVariable* element that has a *component.evse.id*, but *component.evse.connector* is missing	The Charging Station SHALL report the component(s) with this *component.name*, *component.instance* and *component.evse.id* for every *component.evse.connector*, whilst taking into account N02.FR.20.
N02.FR.19	N02.FR.11 AND When Charging Station receives a GetMonitoringReportRequest with a *component* in a *componentVariable* element that has no *component.evse.id*	The Charging Station SHALL report the component(s) with this *component.name*, *component.instance* for every *component.evse* field (including top level component without *component.evse*), whilst taking into account N02.FR.20.
N02.FR.20	N02.FR.11 AND When Charging Station receives a GetMonitoringReportRequest with a *component* in a *componentVariable* element that has a value for *component.instance*	The Charging Station SHALL report the component(s) with this *component.name* for every *component.instance* 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 *component* in a *componentVariable* element that has no *component.instance* field	The Charging Station SHALL report the component(s) with this *component.name* for every *component.instance* field or the component(s) without *component.instance* field, whichever is the case, whilst taking into account N02.FR.18, N02.FR.19.
N02.FR.22 **(2.1)**	N02.FR.03	Charging Station SHALL include the *eventNotificationType* of each monitor in *monitoringData.variableMonitoring*.
N02.FR.23 **(2.1)**	If *monitoringCriteria* contains `TargetDeltaMonitoring`	All monitors with *type* = `TargetDelta` and *type* = `TargetDeltaRelative` are reported.

N03 - Set Monitoring Base

No.	Type	Description
1	**Name**	Set Monitoring Base

No.	Type	Description
2	**ID**	N03
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	**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**.
7	**Error handling**	n/a
8	**Remark(s)**	Upon receipt of a SetMonitoringBaseRequest for: • *monitoringBase* = `HardWiredOnly`: the Charging Station will deactivate all pre-configured monitors and remove any previously configured custom monitors. Only the `HardWiredMonitor` monitors remain. • *monitoringBase* = `FactoryDefault`: the Charging Station will (re)activate all `PreconfiguredMonitor` monitors and remove all custom monitors. • *monitoringBase* = `All`: the Charging Station will activate all pre-configured monitors and leave previously configured `CustomMonitor` monitors intact. This includes the custom monitors that were created when changing an existing pre-configured monitor.

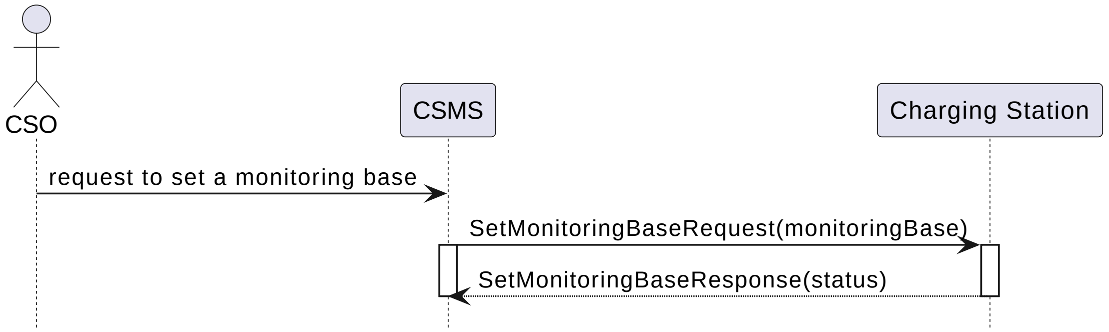

CSO
CSMS
Charging Station
request to set a monitoring base
SetMonitoringBaseRequest(monitoringBase)
SetMonitoringBaseResponse(status)

Figure 167. Sequence Diagram: Set Monitoring Base

N03 - Set Monitoring Base - Requirements

Table 136. N03 - Requirements

ID	Precondition	Requirement definition
N03.FR.01	When the Charging Station accepts a setMonitoringBaseRequest	Then the Charging Station SHALL send a setMonitoringBaseResponse with Accepted.
N03.FR.02	When the Charging Station receives a setMonitoringBaseRequest for a not supported *monitoringBase*	Then the Charging Station SHALL send a setMonitoringBaseResponse with NotSupported.
N03.FR.03	N03.FR.01 AND When the Charging Station received a setMonitoringBaseRequest with *monitoringBase* All	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 setMonitoringBaseRequest with *monitoringBase* FactoryDefault	Then the Charging Station SHALL delete all custom monitors (including overruled pre-configured monitors) and activate the pre-configured monitors of the Charging Station.

ID	Precondition	Requirement definition
N03.FR.05	N03.FR.01 AND When the Charging Station received a setMonitoringBaseRequest with *monitoringBase* HardWiredOnly	Then the Charging Station SHALL clear all custom and disable all pre-configured monitors. Only hard-wired monitors remain active.

N04 - Set Variable Monitoring

No.	Type	Description
1	**Name**	Set Variable Monitoring
2	**ID**	N04
3	**Objective(s)**	To give the CSMS the ability to request the Charging Station to set monitoring triggers on Variables.
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	**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.
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 monitor-able. Otherwise the variableMonitoring setting is removed.

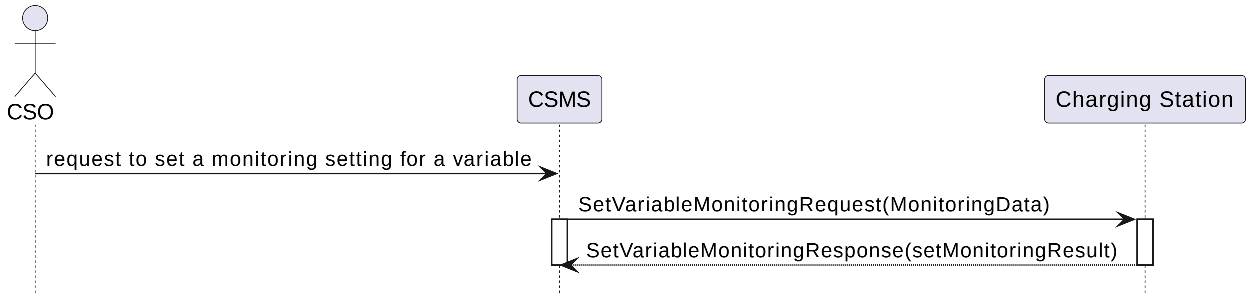

CSO
CSMS
Charging Station
request to set a monitoring setting for a variable
SetVariableMonitoringRequest(MonitoringData)
SetVariableMonitoringResponse(setMonitoringResult)

Figure 168. Sequence Diagram: Set Variable Monitoring

N04 - Set Variable Monitoring - Requirements

Table 137. N04 - Requirements

ID	Precondition	Requirement definition	Note
N04.FR.01	When the Charging Station receives a SetVariableMonitoringRequest with an X number of SetMonitoringData elements	The Charging Station SHALL respond with an SetVariableMonitoringResponse with an equal (X) number of SetMonitoringResult elements, one for every SetMonitoringData element in the SetVariableMonitoringRequest.	
N04.FR.02	N04.FR.01	Every SetMonitoringResult element in the SetVariableMonitoringResponse SHALL contain the same *component* and *variable* combination as one of the SetVariableMonitoringRequest elements in the SetVariableMonitoringRequest.	
N04.FR.03	When the Charging Station receives a SetVariableMonitoringRequest with an unknown Component in SetMonitoringData	The Charging Station SHALL set the *attributeStatus* field in the corresponding SetMonitoringResult to: UnknownComponent.	

ID	Precondition	Requirement definition	Note
N04.FR.04	When the Charging Station receives a SetVariableMonitoringRequest with a Variable that is unknown for the given Component in SetMonitoringData	The Charging Station SHALL set the *attributeStatus* field in the corresponding SetMonitoringResult to: UnknownVariable.	
N04.FR.05	When the Charging Station receives a SetVariableMonitoringRequest with an MonitorType which is not supported by the specific Variable	The Charging Station SHALL set the *attributeStatus* field in the corresponding SetMonitoringResult to: UnsupportedMonitorType.	
N04.FR.06	When the Charging Station receives a SetVariableMonitoringRequest with (monitor *type* = `UpperThreshold` AND *value* < *minLimit* OR *value* > *maxLimit*) OR (monitor *type* = `LowerThreshold` AND *value* < *minLimit* OR *value* > *maxLimit*)	The Charging Station SHALL set the attributeStatus field in the corresponding SetMonitoringResult to: Rejected.	*minLimit* and *maxLimit* refer to the VariableCharacteristicsType for the VariableType. Be aware that setting a `UpperThreshold` to the *maxLimit* or setting a `LowerThreshold` to the *minLimit* will result in a monitor that will never trigger. More information on the reason of rejection can be provided in the optional *statusInfo* element.
N04.FR.07	When the Charging Station receives a SetVariableMonitoringRequest for a monitor that conflicts with safety requirements.	The Charging Station MAY set the *attributeStatus* 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 *value* in the SetMonitoringData	The Charging Station SHALL set the *attributeStatus* 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 *monitoringData* 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 *component/variable* combination for which a monitor with the same *type* and *severity* already exists with a different *id*.	The Charging Station SHALL set the *attributeStatus* 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 UpperThreshold at value "67" and severity "4-Error", then there cannot be another UpperThreshold 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.	

ID	Precondition	Requirement definition	Note
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 attributeStatus 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 attributeStatus field in the corresponding SetMonitoringResult to: Rejected.	More information can be provided in the optional *statusInfo* 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.
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 *value* of 1.	*value* 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 *id* 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

No.	Type	Description
1	**Name**	Set Monitoring Level
2	**ID**	N05
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 limited for some reason. The CSMS can control which events it will to be notified of by the Charging Station with the SetMonitoringLevelRequest message.
	Actors	Charging Station, CSMS, CSO

No.	Type	Description
	Scenario description	**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.
7	**Error handling**	n/a
8	**Remark(s)**	n/a

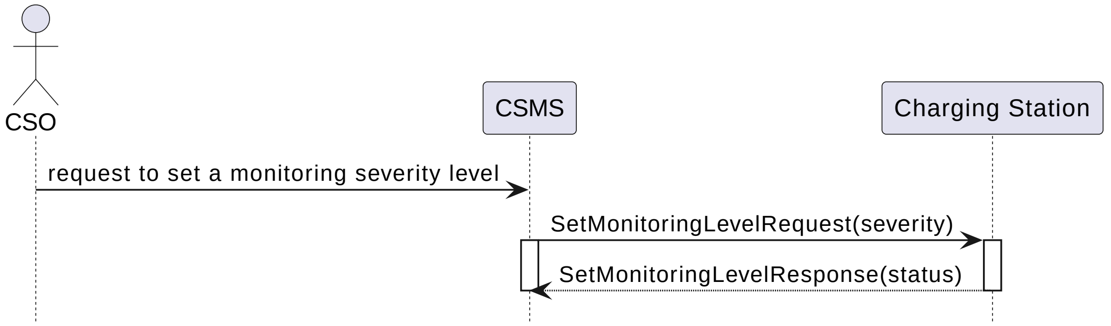

CSO
CSMS
Charging Station
request to set a monitoring severity level
SetMonitoringLevelRequest(severity)
SetMonitoringLevelResponse(status)

Figure 169. Sequence Diagram: Set Monitoring Level

N05 - Set Monitoring Level - Requirements

Table 138. N05 - Requirements

ID	Precondition	Requirement definition
N05.FR.01	When the Charging Station accepts a setMonitoringLevelRequest	The Charging Station SHALL send a setMonitoringLevelResponse with Accepted.
N05.FR.02	When the Charging Station receives a setMonitoringLevelRequest for a *severity* that is out of range	The Charging Station SHALL send a setMonitoringLevelResponse with Rejected.
N05.FR.03	N05.FR.01	The Charging Station SHALL restrict the reporting of monitoring events by NotifyEventRequest to only those monitors with a severity number lower than or equal to the given severity.

N06 - Clear / Remove Monitoring

No.	Type	Description
1	**Name**	Clear / Remove Monitoring
2	**ID**	N06
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 ClearVariableMonitoringRequest with the id of the monitoring setting.
	Actors	Charging Station, CSMS, CSO
	Scenario description	**1.** The CSO triggers the CSMS to request clearing/removing one or more variables in a Charging Station. **2.** The CSMS sends a ClearVariableMonitoringRequest to the Charging Station. **3.** The Charging Station responds with a ClearVariableMonitoringResponse.
5	**Prerequisite(s)**	Charging Station supports Monitoring
6	**Postcondition(s)**	The Charging Station cleared / removed the requested monitoring settings.
7	**Error handling**	n/a
8	**Remark(s)**	n/a

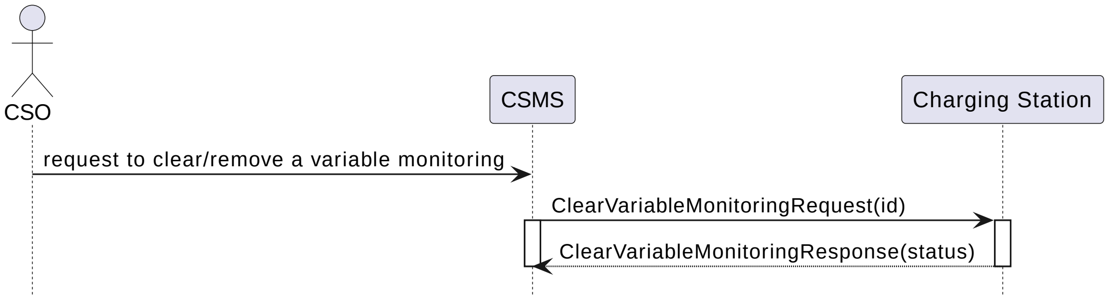

CSO
CSMS
Charging Station
request to clear/remove a variable monitoring
ClearVariableMonitoringRequest(id)
ClearVariableMonitoringResponse(status)

Figure 170. Sequence Diagram: Clear / Remove Monitoring

N06 - Clear / Remove Monitoring - Requirements

Table 139. N06 - Requirements

ID	Precondition	Requirement definition
N06.FR.01	When the Charging Station accepts a ClearVariableMonitoringRequest	The Charging Station SHALL send a ClearVariableMonitoringResponse with Accepted.
N06.FR.02	When the Charging Station receives a ClearVariableMonitoringRequest with a non existing *id*	The Charging Station SHALL send a ClearVariableMonitoringResponse with NotFound.
N06.FR.03	When the Charging Station receives a ClearVariableMonitoringRequest for an *id* 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 *id* elements in a ClearVariableMonitoringRequest than reported by the Charging Station via: `ItemsPerMessageClearVariableMonitoring` and `BytesPerMessageClearVariableMonitoring`.
N06.FR.05		For every *id* 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 *id* elements than allowed by `ItemsPerMessageClearVariableMonitoring`	The Charging Station MAY respond with a CALLERROR(OccurenceConstraintViolation)
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

Updated in OCPP 2.1

No.	Type	Description
1	**Name**	Alert Event
2	**ID**	N07
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

No.	Type	Description
	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.
7	**Error handling**	n/a
8	**Remark(s)**	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. 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 *cleared* 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. 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.

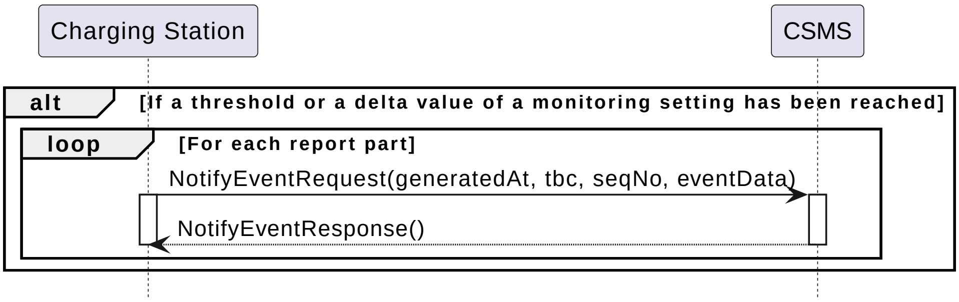

Charging Station
CSMS
alt
[If a threshold or a delta value of a monitoring setting has been reached]
loop
[For each report part]
NotifyEventRequest(generatedAt, tbc, seqNo, eventData)
NotifyEventResponse()

Figure 171. Sequence Diagram: Alert Event

N07 - Alert Event - Requirements

Table 140. N07 - Requirements

ID	Precondition	Requirement definition	Note
N07.FR.02 **(2.1)**	When a monitored value returns to within the boundary of *UpperThreshold*, *LowerThreshold*, *TargetDelta* or *TargetDeltaRelative*	The Charging Station SHALL send a NotifyEventRequest with an eventData with the attribute *cleared* 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 `OfflineMonitoringEventQueuingSeverity` AND The Charging Station is *offline*	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 *eventId* of the other event in the *cause* field of the eventData element in the NotifyEventRequest message.	

ID	Precondition	Requirement definition	Note
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 *seqNo* 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 actualField of the NotifyEventRequest SHALL be empty.	
N07.FR.11 **(2.1)**	When modifying a set *UpperThreshold*, *LowerThreshold*, *TargetDelta* or *TargetDeltaRelative* VariableMonitor AND (the new threshold clears the old threshold OR the new threshold is exceeded by the monitored value)	The Charging Station SHALL send a NotifyEventRequest message for this VariableMonitor with the appropriate value for *cleared*.	If monitored value is below new threshold, then *cleared* = true. If monitored value above threshold, then *cleared* = false or absent.
N07.FR.12 **(2.1)**	When removing a set *UpperThreshold*, *LowerThreshold*, *TargetDelta* or *TargetDeltaRelative* VariableMonitor AND the threshold was exceeded.	The Charging Station SHALL NOT send a NotifyEventRequest with an eventData with the attribute *cleared* is true.	
N07.FR.13		A VariableMonitoring needs to be stored persistently across reboots.	
N07.FR.14 **(2.1)**	When a variableMonitoring setting of type *UpperThreshold*, *LowerThreshold*, *TargetDelta* or *TargetDeltaRelative* 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 *cleared* is true.	
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 *value*	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 *value*	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 *value* 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.	

ID	Precondition	Requirement definition	Note
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 *value*)	The Charging Station SHALL send a NotifyEventRequest with *trigger* `Delta` for the triggered monitor.	
N07.FR.20 **(2.1)**	N07.FR.06	Charging Station SHOULD set the field *eventData.severity* of the NotifyEventRequest to the *severity* of the monitor referred to in *variableMonitoringId*.	A HardwiredNotification does not have an associated monitor, but can still provide a *severity*. The requirement is a "should", because the added field is optional for backwards compatibility, but it is recommended to provide a severity.
N07.FR.21 **(2.1)**	When *eventData.eventNotificationType* = `HardwiredNotification` in NotifyEventRequest	Charging Station SHOULD set *eventData.severity* to an implementation-defined value.	Value of severity is up to implementation, since this notification is not related to a monitor.
N07.FR.22 **(2.1)**	When there is a monitor with type `TargetDelta` on a Component/Variable combination AND the Actual value (*attributeType* `Actual`) of the Variable differs from the Target value (*attributeType* `Target`) more than plus or minus *value*	The Charging Station SHALL send a NotifyEventRequest with trigger `Alerting` for the triggered monitor.	Example: when *target* = 100, *value* = 10, then an event is triggered when *actual* < 90 or *actual* > 110.
N07.FR.23 **(2.1)**	When there is a monitor with type `TargetDeltaRelative` on a Component/Variable combination AND the Actual value (*attributeType* `Actual`) of the Variable differs from the Target value (*attributeType* `Target`) more than plus or minus (*value* * Target value)	The Charging Station SHALL send a NotifyEventRequest with trigger `Alerting` for the triggered monitor.	Example: when *target* = 100, *value* = 0.1, then an event is triggered when *actual* < 90 or *actual* > 110.

N08 - Periodic Event

No.	Type	Description
1	**Name**	Periodic Event
2	**ID**	N08
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.
7	**Error handling**	n/a
8	**Remark(s)**	n/a

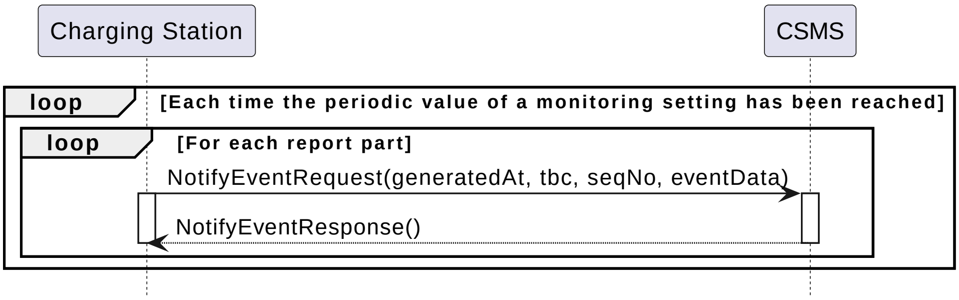

Charging Station
CSMS
loop
[Each time the periodic value of a monitoring setting has been reached]
loop
[For each report part]
NotifyEventRequest(generatedAt, tbc, seqNo, eventData)
NotifyEventResponse()

Figure 172. Sequence Diagram: Periodic Event

N08 - Periodic Event - Requirements

Table 141. 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 *offline*	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 *seqNo* 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 *trigger* 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 *value* 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 *value*, starting from the nearest clock-aligned interval after this monitor was set, have passed (For example, a *value* 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

No.	Type	Description
1	**Name**	Get Customer Information
2	**ID**	N09
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	**1.** The CSMS sends a CustomerInformationRequest with the report flag set to *true* to the Charging Station with a reference to a customer (idToken, customerCertificate or customerIdentifier). **2.** The Charging Station responds with CustomerInformationResponse, indicating whether it will send it or not. **3.** The Charging Station sends one or more NotifyCustomerInformationRequest messages to the CSMS. **4.** The CSMS responds with one or more NotifyCustomerInformationResponse messages to the Charging Station.
5	**Prerequisite(s)**	n/a
6	**Postcondition(s)**	The CSMS has *Successfully* received a CustomerInformationResponse message with status *Accepted* AND has *Successfully* received the requested data.
7	**Error handling**	n/a
8	**Remark(s)**	n/a

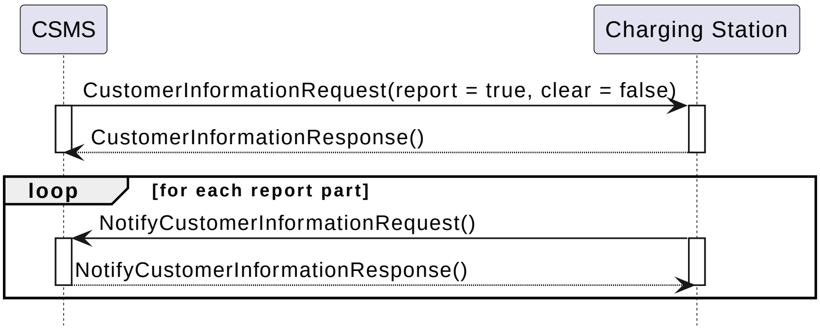

CSMS
Charging Station
CustomerInformationRequest(report = true, clear = false)
CustomerInformationResponse()
loop
[for each report part]
NotifyCustomerInformationRequest()
NotifyCustomerInformationResponse()

Figure 173. Sequence Diagram: Get Customer Information

N09 - Get Customer Information - Requirements

Table 142. 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 *true*.	
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 *Accepted* .	
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 *Rejected* .	
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 *false*	It is RECOMMENDED to respond with status a CustomerInformationResponse message with status *Rejected* .	
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 GetInstalledCertificateIdsRequest to be sure of the used *hashAlgorithm*.
N09.FR.09	When CustomerInformationRequest contains none of *idToken*, *customerCertificate* or *customerIdentifier* OR CustomerInformationRequest contains more than one of *idToken*, *customerCertificate* or *customerIdentifier*	Charging Station SHALL respond with *status* = `Invalid`	Only one value for either *idToken*, *customerCertificate* or *customerIdentifier* may be provided. Charging Station counterpart requirement of N09.FR.04.

N10 - Clear Customer Information

No.	Type	Description
1	**Name**	Clear Customer Information
2	**ID**	N10
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	**1.** The CSMS sends CustomerInformationRequest with the clear flag set to *true* to the Charging Station with a reference to a customer (idToken, customerCertificate or customerIdentifier). **2.** The Charging Station responds with CustomerInformationResponse, indicating whether it will send it or not. **3.** If the report flag is set to *true*, the Charging Station sends one or more NotifyCustomerInformationRequest messages to the CSMS. **4.** The CSMS responds with one or more NotifyCustomerInformationResponse messages to the Charging Station.
5	**Prerequisite(s)**	n/a
6	**Postcondition(s)**	The CSMS has *Successfully* received a CustomerInformationResponse message with status *Accepted*, the Charging Station has removed the customer information as requested and (if report flag was set to *true*) the CSMS has *Successfully* received the removed data.
7	**Error handling**	n/a
8	**Remark(s)**	n/a

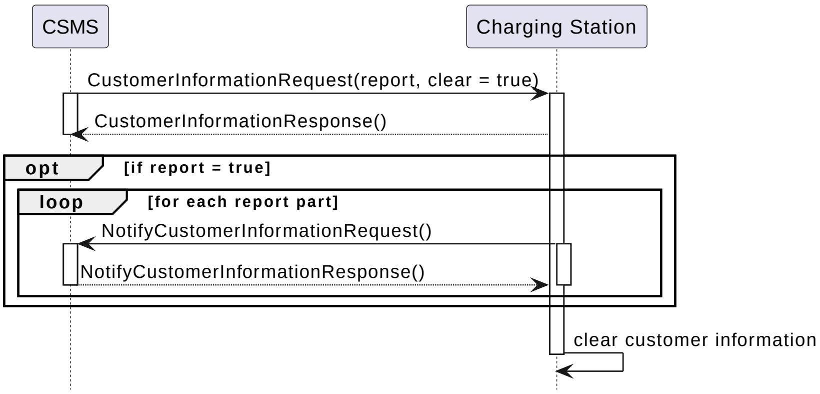

CSMS
Charging Station
CustomerInformationRequest(report, clear = true)
CustomerInformationResponse()
opt
[if report = true]
loop
[for each report part]
NotifyCustomerInformationRequest()
NotifyCustomerInformationResponse()
clear customer information

Figure 174. Sequence Diagram: Clear Customer Information

N10 - Clear Customer Information - Requirements

Table 143. 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 *Accepted* .	
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 *true* and the report flag set to *true* 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 *true* and the report flag set to *true* 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 *Rejected*	
N10.FR.06	N10.FR.01 AND receiving a CustomerInformationRequest with the clear flag set to *true*, the report flag set to *false*	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 *false*	It is RECOMMENDED to respond with a CustomerInformationResponse message with status *Rejected* .	
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 *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 GetInstalledCertificateIdsRequest to be sure of the used *hashAlgorithm*.

2.5. Frequent Periodic Variable Monitoring

The following use cases describe a more efficient method of reporting values of periodic monitors that need to be reported frequently. A Charging Station opens a so-called "event stream" for a specific monitor. This can be at the request of CSMS in a SetVariableMonitoringRequest, or by Charging Station when it is reporting periodic values for a preconfigured or hardwired monitor.

Charging Station may send these periodic values in chunks to safe overhead. For example, a periodic monitor of 1 second can be reported in one NotifyPeriodicEventStream message that is sent every minute with 60 values.

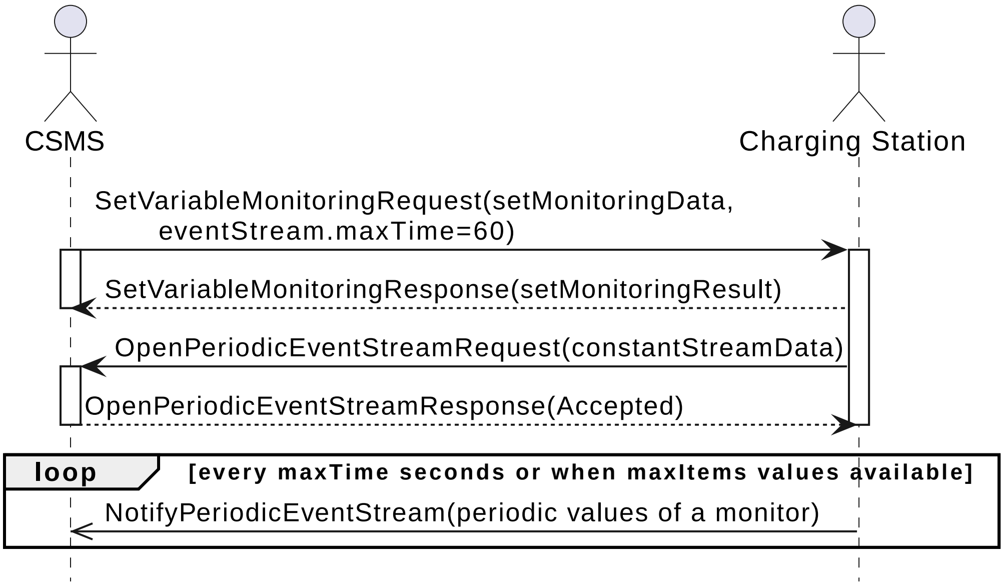
The NotifyPeriodicEventStream message is of a special unconfirmed message type ("SEND") that requires no response. This allows for much faster sending of the data.

N11 - Set Frequent Periodic Variable Monitoring

New in OCPP 2.1

No.	Type	Description
1	**Name**	Set Frequent Periodic Variable Monitoring
2	**ID**	N11
3	**Objective(s)**	To give the CSMS the ability to request efficient frequent periodic monitoring of variables.
4	**Description**	This use case describes how the CSMS requests the Charging Station to set monitoring triggers on Variables and use an event stream to report a set of measured values, for example, once every 60 seconds to reduce the communications overhead of individual NotifyEventRequests.
	Actors	Charging Station, CSMS, CSO
	Scenario description #1	*Variable monitor set by CSMS* 1. The CSMS sends a SetVariableMonitoringRequest for two variables of a component to the Charging Station with *monitorEnumType* = `Periodic` or `PeriodicClockAligned` and with *eventStream.interval/values* describing how often the stream must be flushed. 2. Charging Station responds with a SetVariableMonitoringResponse with *setMonitoringResult* element for each variable and an `Accepted` status and an *id* for each monitor. 3. Charging Station creates an event stream for each monitored variable by sending an OpenPeriodicEventStreamRequest to CSMS with the constant data for each variable for which an event stream is created. 4. CSMS responds to each OpenPeriodicEventStreamRequest with OpenPeriodicEventStreamResponse with *status* = `Accepted`. 5. Every *interval* seconds or when *values* values are available Charging Station will send all periodic values of a monitor as one message via the NotifyPeriodicEventStream message. This message is not acknowledged by CSMS.
	Scenario description #2	*Hardwired/PreconfiguredMonitor from Charging Station* 1. Charging Station creates an event stream for each periodic HardWiredMonitor or PreconfiguredMonitor by sending an OpenPeriodicEventStreamRequest to CSMS with the constant data for each variable for which an event stream is created and *params.interval/values* describing how often the stream will be flushed. 2. CSMS responds to each OpenPeriodicEventStreamRequest with OpenPeriodicEventStreamResponse with *status* = `Accepted`. 3. Every *interval* seconds or when *values* values are available Charging Station will send all periodic values of a monitor as one message via the NotifyPeriodicEventStream message. This message is not acknowledged by CSMS.
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.
7	**Error handling**	If Charging Station fails to open an event stream, e.g. because CSMS rejects it, then Charging Station will use regular NotifyEventRequest messages.

No.	Type	Description
8	**Remark(s)**	This mechanism is meant for monitors that are expected to be triggered frequently, like periodic monitors, but it may also be used for delta monitors that are expected to trigger frequently. 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 monitor-able. Otherwise, the variableMonitoring setting is removed. A Charging Station can also open an event stream for hardwired or preconfigured monitors, for which no SetVariableMonitoringRequest was issued.


CSMS
Charging Station
SetVariableMonitoringRequest(setMonitoringData, eventStream.maxTime=60)
SetVariableMonitoringResponse(setMonitoringResult)
OpenPeriodicEventStreamRequest(constantStreamData)
OpenPeriodicEventStreamResponse(Accepted)
loop
[every maxTime seconds or when maxItems values available]
NotifyPeriodicEventStream(periodic values of a monitor)

N11 - Set Frequent Periodic Variable Monitoring - Requirements

ID	Precondition	Requirement definition
N11.FR.01	Upon receiving a SetVariableMonitoringRequest with one or more *setMonitoringData* elements that have a *periodicEventStream* element	Charging Station SHALL associate a periodic event stream with each of the monitors that have a *periodicEventStream* element.
N11.FR.02	N11.FR.01	Charging Station SHALL send a OpenPeriodicEventStreamRequest to CSMS with the *id* of event stream, and *params* set to the event stream flushing parameters from *periodicEventStream* in the SetVariableMonitoringRequest, and *variableMonitoringId* set to the id of the monitor.
N11.FR.03	When Charging Station wants to use a periodic event stream for a HardWired or Preconfigured monitor	Charging Station SHALL send a OpenPeriodicEventStreamRequest to CSMS with the *id* of event stream, and *params* set to the (preconfigured) event stream flushing parameters, and *variableMonitoringId* set to the id of the monitor.
N11.FR.04	N11.FR.02 OR N11.FR.03	CSMS SHALL prepare to receive data of the periodic event stream and SHALL associate the data related to the *variableMonitoringId* with the values read from the event stream.
N11.FR.05	When CSMS is not able to process the periodic event stream	CSMS SHALL respond with a OpenPeriodicEventStreamResponse with *status* = `Rejected`.
N11.FR.06	When CSMS accepts the periodic event stream	CSMS SHALL respond with a OpenPeriodicEventStreamResponse with *status* = `Accepted`.
N11.FR.07	N11.FR.05	Charging Station SHALL refrain from using the periodic event stream for this monitor and SHALL revert to using NotifyEventRequest message for this monitor.

ID	Precondition	Requirement definition
N11.FR.08		CSMS SHALL associate the following data, that is normally present as required fields in a NotifyEventRequest, with the periodic event stream. (This data is available from the monitor): • *variableMonitoringId*, • *trigger* (= `Periodic`), • *eventNotificationType*, • *severity*, • *component*, • *variable*.
N11.FR.09	When providing a *periodicEventStream* element in a SetVariableMonitoringRequest	CSMS SHALL provide a value for *interval* or *values* or both.

N12 - Get Periodic Event Streams

New in OCPP2.1

No.	Type	Description
1	**Name**	Get Periodic Event Streams
2	**ID**	N12
3	**Objective(s)**	To get a list of existing event streams
4	**Description**	CSMS requests a list of event streams and associated constant data.
	Actors	Charging Station, CSMS, CSO
	Scenario description	1. The CSMS sends a GetPeriodicEventStreamRequest without parameters. 2. CS responds with a GetPeriodicEventStreamResponse with a list of ConstantStreamDataType.
5	**Prerequisite(s)**	
6	**Postcondition(s)**	
7	**Error handling**	
8	**Remark(s)**	This is only needed in case CSMS has somehow lost its administration of event streams.

N12 - Get Periodic Event Streams - Requirements

ID	Precondition	Requirement definition
N12.FR.01	Upon receiving a GetPeriodicEventStreamRequest	Charging Station SHALL respond with a GetPeriodicEventStreamResponse with a list of zero or more ConstantStreamDataType of all open event streams

N13 - Close Periodic Event Streams

New in OCPP2.1

No.	Type	Description
1	**Name**	Close Periodic Event Streams
2	**ID**	N13
3	**Objective(s)**	To close a periodic event stream
4	**Description**	Charging Station closes/terminates a periodic event stream.
	Actors	Charging Station, CSMS, CSO

No.	Type	Description
	Scenario description #1	*Charging Station closes a periodic event stream* 1. Charging Station flushes the periodic event stream, i.e. all buffered data is sent to CSMS. 2. Charging Station reverts to using regular NotifyEventRequest messages for the variable that was associated with the periodic event stream. 3. Charging Station sends a ClosePeriodicEventStreamRequest with *id* of the event stream to tell CSMS that event stream has been closed. 4. CSMS responds with a ClosePeriodicEventStreamResponse.
	Scenario description #2	*CSMS clears a variable monitor* 1. CSMS sends a ClearVariableMonitoringRequest for a component/variable monitor that uses a periodic event stream to report the events. 2. Charging Station processes the request as described in use case N06 Clear/Remove Monitoring. 3. If monitor was succesfully cleared, Charging Station continues with steps from *Scenario description #1*.
5	**Prerequisite(s)**	
6	**Postcondition(s)**	
7	**Error handling**	
8	**Remark(s)**	This can be initiated by Charging Station at any time, but is normally the result of CSMS removing a monitor.

N13 - Close Periodic Event Streams - Requirements

ID	Precondition	Requirement definition	Note
N13.FR.01	When Charging Station wants to close/terminate a periodic event stream	Charging Station SHALL notify CSMS about this by sending a ClosePeriodicEventStreamRequest message with the *id* of the event stream.	
N13.FR.02	N13.FR.01	CSMS SHALL stop using the periodic event stream and respond with a ClosePeriodicEventStreamResponse.	
N13.FR.03	N13.FR.01	Charging Station SHALL send all buffered values of a periodic event stream before sending the ClosePeriodicEventStreamRequest.	
N13.FR.04	N13.FR.01	Charging Station SHALL revert to using NotifyEventRequest message to report the variable that was associated with the periodic event stream.	
N13.FR.05	Upon receiving a ClearVariableMonitoringRequest for a monitor	Charging Station SHALL clear the monitor as described in use case N06 Clear/Remove Monitor, and Charging Station SHALL close an associated periodic event stream as per requirement N13.FR.01.	Removing the monitor automatically closes the periodic event stream.
N13.FR.06	Upon receiving a SetVariableMonitoringRequest for an existing monitor with same *id* and same component/variable and without a *periodicEventStream* element	Charging Station SHALL replace the monitor as described in N04.FR.12, and Charging Station SHALL close an associated periodic event stream as per requirement N13.FR.01.	This re-installs the monitor without a periodic event stream

N14 - Adjust Periodic Event Streams

New in OCPP2.1

No.	Type	Description
1	**Name**	Adjust Periodic Event Streams

No.	Type	Description
2	**ID**	N14
3	**Objective(s)**	To adjust the transmission rate of a periodic event stream
4	**Description**	CSMS updates the *interval/values* parameters of an event stream.
	Actors	Charging Station, CSMS
	Scenario description #1	1. CSMS sends a AdjustPeriodicEventStreamRequest to Charging Station with new values for *params.interval* and/or *params.values* to change how often data for the stream is sent.
5	**Prerequisite(s)**	A periodic event stream exists.
6	**Postcondition(s)**	How often Charging Station sends data for the periodic event stream is adjusted accordingly.
7	**Error handling**	
8	**Remark(s)**	

N14 - Adjust Periodic Event Streams - Requirements

ID	Precondition	Requirement definition	Note
N14.FR.01	When CSMS wants to change the rate of sending data for a periodic event stream	CSMS SHALL send a AdjustPeriodicEventStreamRequest message with the *id* of the event stream and a *params* object with new *interval* and/or *values* fields	
N14.FR.02	N14.FR.01 AND If Charging Station is unable to comply to the new parameters	Charging Station SHALL respond with AdjustPeriodicEventStreamResponse with *status* = `Rejected`.	
N14.FR.03	N14.FR.01 AND If Charging Station is able to comply to the new parameters	Charging Station SHALL respond with AdjustPeriodicEventStreamResponse with *status* = `Accepted` and change the rate of sending NotifyPeriodicEventStream messages according to *params*.	

N15 - Periodic Event Streams

New in OCPP2.1

No.	Type	Description
1	**Name**	Periodic Event Streams
2	**ID**	N15
3	**Objective(s)**	To explain how a frequent periodic monitor is sent.
4	**Description**	Charging Station can open a periodic event stream to report frequent periodic monitor values, as described in use case N11. This reduces data overhead, since static data is not repeated every time, and reduces time, because a periodic event stream message is not acknowledged by CSMS. This use case describes the use of the NotifyPeriodicEventStream message to send the data.
	Actors	Charging Station, CSMS
	Scenario description #1	*Charging Station sends values over event stream* 1. When a monitor that is associated with a periodic event stream, produces a value, Charging Station places this value in the event stream buffer together with a timestamp. 2. The field *params* of PeriodicEventStreamParamsType that was used in OpenPeriodicEventStreamRequest determines when to flush the buffer. 3. After more than *params.interval* seconds have passed since the last flush, or after the number of values equals or exceeds *params.values*, then Charging Station writes ("flushes") the buffered values to CSMS as a list of *streamData* in a NotifyPeriodicEventStream message. This message is not confirmed by CSMS.

No.	Type	Description
	Scenario description #2	*CSMS reads values from event stream* 1. When CSMS receives a NotifyPeriodicEventStream message, it has a *basetime* and a list of StreamDataElementType with the value (*v*) and time (*t*), as an offset to *basetime*, of the event. 2. CSMS combines the data with the information of the monitor of this periodic event stream such that it can recreate all required fields that would be present in a NotifyEventRequest. This allows a CSMS to use the same processing for events from a periodic event stream as for regular NotifyEventRequests. **NOTE** One required field of NotifyEventRequest is missing: *eventId*, because that is not present in the periodic event stream data. When needed CSMS can give it any value, because it is never communicated back to the Charging Station.
5	**Prerequisite(s)**	A periodic event stream for the monitor exists.
6	**Postcondition(s)**	
7	**Error handling**	Since this message has no reply, there is no way for CSMS to report an error condition back to Charging Station. The only way for CSMS to stop with a periodic event stream is by clearing the associated monitor.
8	**Remark(s)**	This message uses the RPC framework message type "SEND" to send an unconfirmed message. Unconfirmed messages do not require a response. They were introduced in OCPP 2.1. (See part 4 "JSON over Websockets Implementation Guide"). A Charging Station can send a SEND message at any time, i.e. there is need to wait for the CALLRESULT of the previous CALL message before sending it. Because SEND messages still use the same websocket connection as CALL/CALLRESULT messages, the frequent sending of large SEND messages may cause a delay for other messages.

N15 - Periodic Event Stream - Requirements

ID	Precondition	Requirement definition
N15.FR.01		Charging Station SHALL send a NotifyPeriodicEventStream message as a message type "SEND" in the RPC Framework over the websocket.
N15.FR.02	When receiving a message of RPC message type "SEND"	CSMS SHALL NOT confirm the message upon receipt with "CALLRESULT" or "CALLERROR".
N15.FR.03	N15.FR.02	CSMS SHALL associate the following fixed data, that is normally present as required fields in a NotifyEventRequest, with the *data* elements of the periodic event stream of type StreamDataElementType. (This fixed data is available from the monitor in *variableMonitoringId*): • *variableMonitoringId*, • *trigger* (= `Periodic`), • *eventNotificationType*, • *severity*, • *component*, • *variable*.
N15.FR.04		Charging Station SHALL set *pending* in NotifyPeriodicEventStream to the number of *data* elements that is available, but not yet part of this NotifyPeriodicEventStream message.
N15.FR.05	When the value of *pending* in subsequent NotifyPeriodicEventStream messages that CSMS receives is growing	CSMS SHOULD consider adjusting the periodic event stream parameters to allow larger or more frequent messages.
N15.FR.06	If *params.interval* in a SetMonitoringDataType of a monitor is greater than 0	Charging Station SHALL send a NotifyPeriodicEventStream message for the associated monitor every *params.interval* seconds.

ID	Precondition	Requirement definition
N15.FR.07	N15.FR.06 AND *params.values* is greater than 0	Charging Station SHALL send no more than *params.values* values at a time in a NotifyPeriodicEventStream message.
N15.FR.08	If *params.values* in a SetMonitoringDataType of a monitor is greater than 0	Charging Station SHALL send a NotifyPeriodicEventStream message for the associated monitor as soon as *params.values* values are available.
N15.FR.09		Charging Station SHALL set *basetime* in a NotifyPeriodicEventStream message to the timestamp of the first value in the *data* list of values. (This effectively makes that the offset *t* of the first value is always 0).

O. Display Message

Chapter 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.)

Chapter 2. Use cases & Requirements

O01 - Set DisplayMessage

Updated in OCPP 2.1

No.	Type	Description
1	**Name**	Set DisplayMessage
2	**ID**	O01
3	**Objective(s)**	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	**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)	O02 - Set DisplayMessage for Transaction O06 - 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)
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. The default language of the Charging Station and the list of supported languages are reported in the Configuration Variable: `DisplayMessageLanguage`. If `DisplayMessageLanguage` is not present, then additional language messages in *messageExtra* in a SetDisplayMessageRequest are not supported.

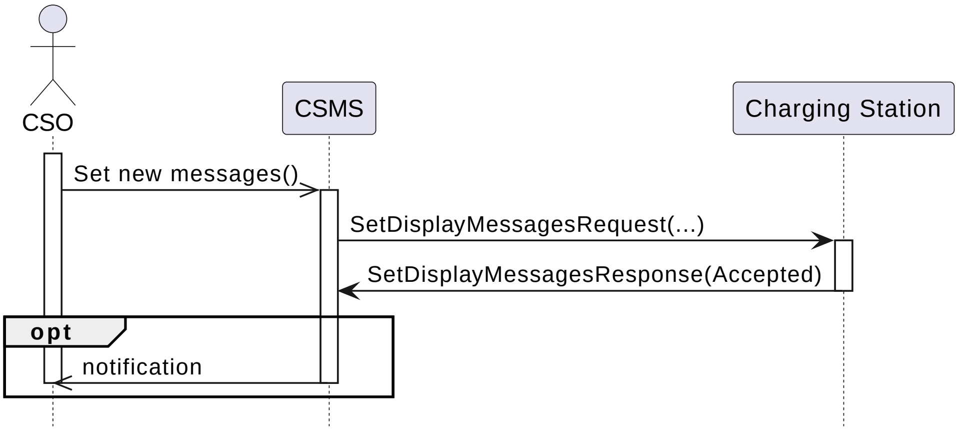

CSO
CSMS
Charging Station
Set new messages()
SetDisplayMessagesRequest(...)
SetDisplayMessagesResponse(Accepted)
opt
notification

Figure 175. Set DisplayMessage sequence diagram

O01 - Set DisplayMessage - Requirements

Table 144. O01 - Requirements

ID	Precondition	Requirement definition
O01.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**. (See configuration variable DisplayMessageSupportedPriorities for supported priorities).
O01.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**. (See configuration variable DisplayMessageSupportedStates for supported states).
O01.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**. (See configuration variable DisplayMessageSupportedFormats for supported formats).
O01.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.
O01.FR.05		The CSMS MAY include a startTime and endTime when setting a message.
O01.FR.06	O01.FR.05	The Charging Station SHALL NOT display the DisplayMessage message before the startTime.
O01.FR.07	O01.FR.05	The Charging Station SHALL remove a DisplayMessage message after the endTime.
O01.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.
O01.FR.09	O01.FR.08	When no matching language is available, it is RECOMMENDED to show a DisplayMessage message in English as fall-back, if available.
O01.FR.10		The Charging Station SHALL store the messages in persistent storage, so they survive a power cycle/reboot of the Charging Station.
O01.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.
O01.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.
O01.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.
O01.FR.14	When multiple messages with priority *InFront* are configured to be shown at the same time	The Charging Station SHALL cycle these messages.
O01.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.
O01.FR.16	O01.FR.15 AND Another message with priority *AlwaysFront* is already set	The Charging Station SHALL replace the old message with the newly set message.
O01.FR.17		Language SHALL be specified as RFC-5646 tags, see: [RFC5646], example: US English is: "en-US"
O01.FR.18 **(2.1)**	If configuration variable DisplayMessageLanguage is present AND Charging Station receives a SetDisplayMessageRequest with a *message* or *messageExtra* element in its MessageInfoType that has a *language* that is not in the *valuesList* of DisplayMessageLanguage	Charging Station SHALL respond with a SetDisplayMessageResponse with *status* = `LanguageNotSupported`.

ID	Precondition	Requirement definition
O01.FR.19 **(2.1)**	If configuration variable DisplayMessageLanguage is not present AND Charging Station receives a SetDisplayMessageRequest with a *messageExtra* element in its MessageInfoType	Charging Station SHALL respond with a SetDisplayMessageResponse with *status* = `Rejected`.

O02 - Set DisplayMessage for Transaction

No.	Type	Description
1	**Name**	Set DisplayMessage for Transaction
2	**ID**	O02
	Parent use case	O01 - Set DisplayMessage
3	**Objective(s)**	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	**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)	O01 - Set MessageMessage O06 - 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)
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.

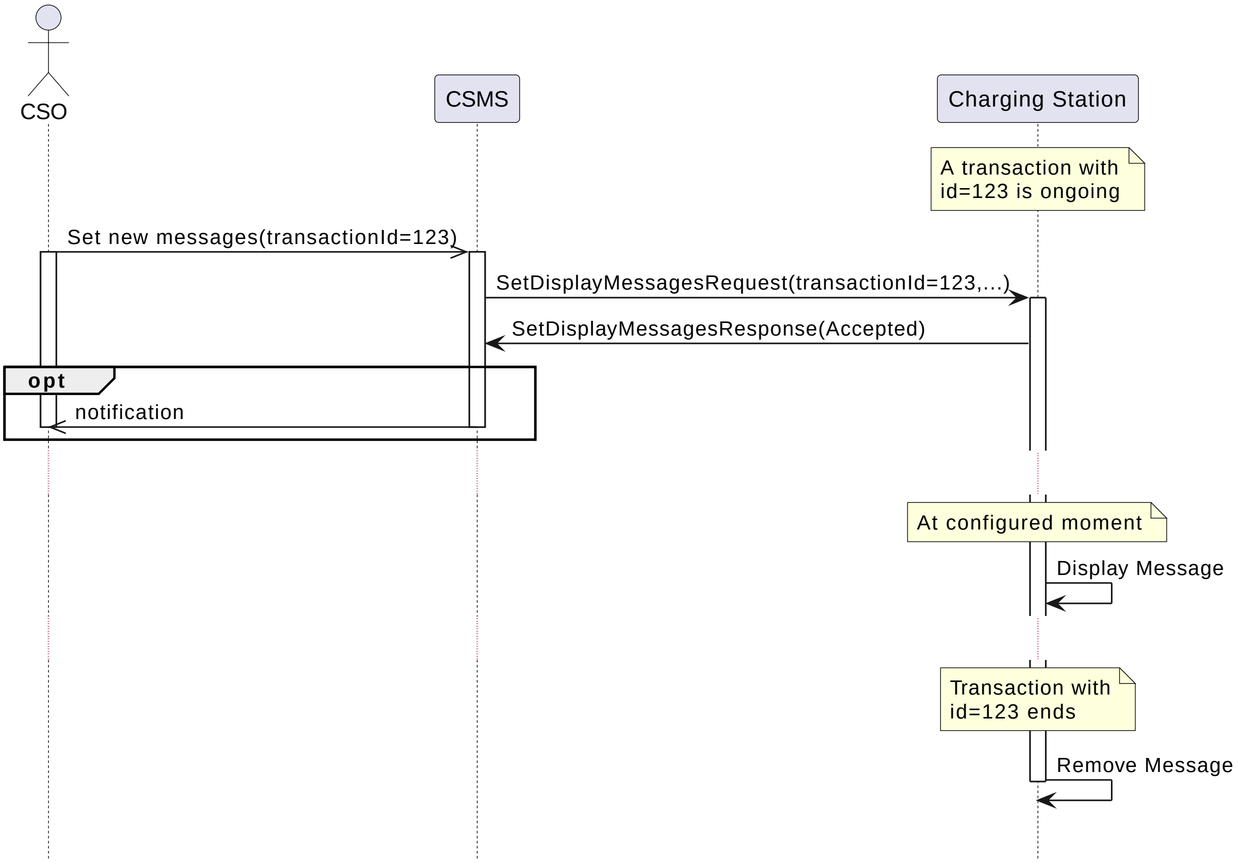

CSO
CSMS
Charging Station
A transaction with id=123 is ongoing
Set new messages(transactionId=123)
SetDisplayMessagesRequest(transactionId=123,...)
SetDisplayMessagesResponse(Accepted)
opt
notification
At configured moment
Display Message
Transaction with id=123 ends
Remove Message

Figure 176. Set DisplayMessage for transaction sequence diagram

O02 - Set DisplayMessage for Transaction - Requirements

Table 145. O02 - Requirements

ID	Precondition	Requirement definition
O02.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**.
O02.FR.02	When the transaction with the given **transactionId** ends	The Charging Station SHALL remove the message from the list of messages.
O02.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**.
O02.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**.
O02.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**.
O02.FR.06		The Charging Station SHALL NOT display the DisplayMessage message before the startTime.
O02.FR.07		The Charging Station SHALL remove a DisplayMessage message after the endTime.
O02.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.
O02.FR.09	O02.FR.08	When no matching language is available, it is RECOMMENDED to show a DisplayMessage message in English as fall-back, if available.
O02.FR.10		The Charging Station SHALL store the messages in persistent storage, so they survive a power cycle/reboot of the Charging Station.
O02.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.
O02.FR.12		Language SHALL be specified as RFC-5646 tags, see: [RFC5646], example: US English is: "en-US"
O02.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.
O02.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.
O02.FR.16	When multiple messages with priority *InFront* are configured to be shown at the same time	The Charging Station SHALL cycle these messages.
O02.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 messaged. Hence, it shall not cycle it with other messages and the Charging Station's own message shall not override this message.
O02.FR.18	O02.FR.17 AND Another message with priority *AlwaysFront* is already set	The Charging Station SHALL replace the old message with the newly set message.

O03 - Get All DisplayMessages

No.	Type	Description
1	**Name**	Get All DisplayMessages
2	**ID**	O03
3	**Objective(s)**	Enable a CSO to retrieve all messages currently configured in a Charging Station.
4	**Description**	This use case describes how a CSO can request all the installed DisplayMessages configured via OCPP in a Charging Station. 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 to current list of messages, so the CSO knows which messages are (still) configured.
	Actors	CSO, CSMS, Charging Station
	Scenario description	**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
7	**Error Handling**	n/a
8	**Remarks**	Only messages configured via OCPP can be retrieved via a GetDisplayMessagesRequest.

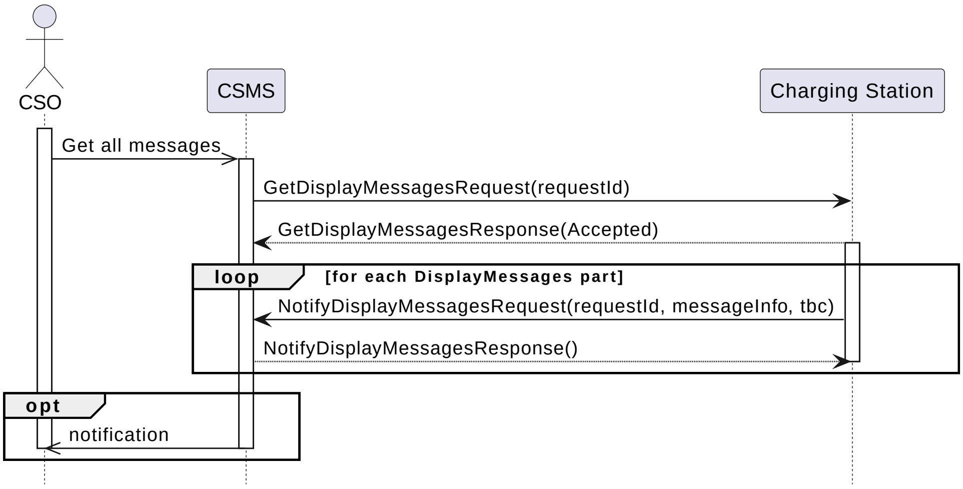

CSO
CSMS
Charging Station
Get all messages
GetDisplayMessagesRequest(requestId)
GetDisplayMessagesResponse(Accepted)
loop
[for each DisplayMessages part]
NotifyDisplayMessagesRequest(requestId, messageInfo, tbc)
NotifyDisplayMessagesResponse()
opt
notification

Figure 177. Get All DisplayMessages sequence diagram

O03 - Get All DisplayMessages - Requirements

Table 146. O03 - Requirements

ID	Precondition	Requirement definition
O03.FR.01	When all fields except *requestId* in a GetDisplayMessagesRequest are omitted AND at least one display message is configured.	The Charging Station SHALL respond with *Accepted*.
O03.FR.02	O03.FR.01	The Charging Station SHALL send all configured DisplayMessages via NotifyDisplayMessagesRequest.
O03.FR.03	O03.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.
O03.FR.04	O03.FR.03	The Charging Station SHALL set the *tbc* field is *true* in every NotifyDisplayMessagesRequest messages, except the last.

ID	Precondition	Requirement definition
O03.FR.05	O03.FR.04	The Charging Station SHALL set the *requestId* field to the same value as the *requestId* in the GetDisplayMessagesRequest.
O03.FR.06	When NO DisplayMessages are configured	The Charging Station SHALL respond with *Unknown*.

O04 - Get Specific DisplayMessages

No.	Type	Description
1	**Name**	Get Specific DisplayMessages
2	**ID**	O04
3	**Objective(s)**	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 out-dated, 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	**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
7	**Error Handling**	n/a
8	**Remarks**	Only message configured via OCPP can be retrieved via GetDisplayMessagesRequest.

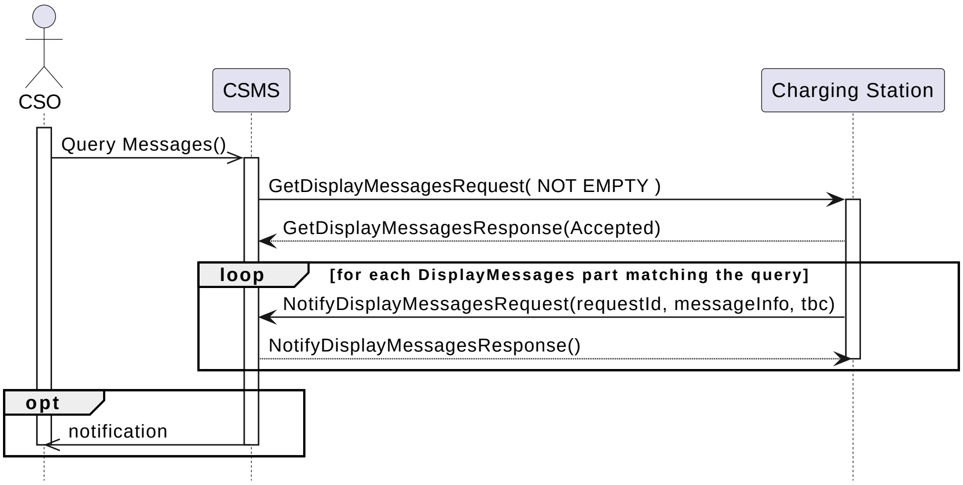

CSO
CSMS
Charging Station
Query Messages()
GetDisplayMessagesRequest(NOT EMPTY)
GetDisplayMessagesResponse(Accepted)
loop
[for each DisplayMessages part matching the query]
NotifyDisplayMessagesRequest(requestId, messageInfo, tbc)
NotifyDisplayMessagesResponse()
opt
notification

Figure 178. Get a specific DisplayMessages sequence diagram

O04 - Get Specific DisplayMessage - Requirements

Table 147. O04 - Requirements

ID	Precondition	Requirement definition
O04.FR.01	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	The Charging Station SHALL respond with *Accepted*.
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 *Unknown*.

ID	Precondition	Requirement definition
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 *tbc* field is *true* in every NotifyDisplayMessagesRequest messages, except the last.
O04.FR.06	O04.FR.05	The Charging Station SHALL set the *requestId* field to the same value as the *requestId* in the GetDisplayMessagesRequest.
O04.FR.07	When NO DisplayMessages are configured	The Charging Station SHALL respond with *Unknown*.

O05 - Clear a DisplayMessage

No.	Type	Description
1	**Name**	Clear a DisplayMessage
2	**ID**	O05
3	**Objective(s)**	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	**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
7	**Error Handling**	n/a
8	**Remarks**	Only messages configured via OCPP can be cleared/removed via ClearDisplayMessageRequest

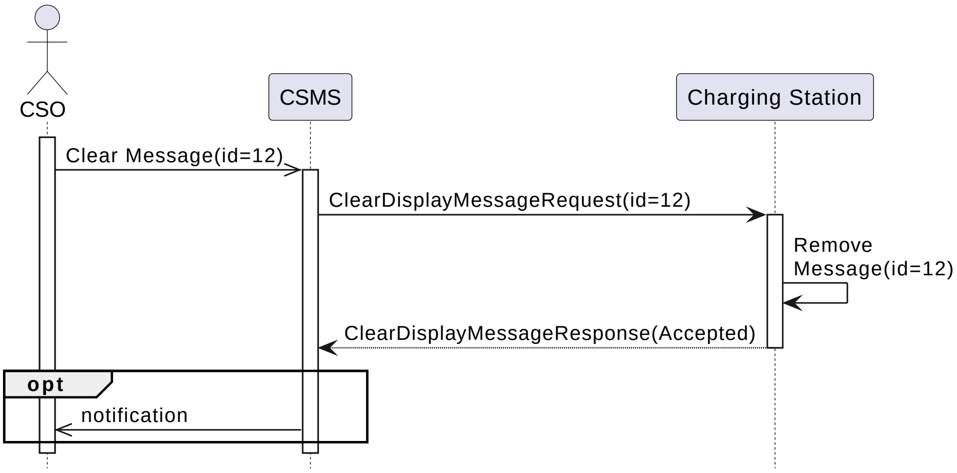


Figure 179. Clear a DisplayMessage sequence diagram

O05 - Clear a DisplayMessage - Requirements

Table 148. O05 - Requirements

ID	Precondition	Requirement definition
O05.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*.
O05.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*.
O05.FR.03 **(2.1)**	When a Charging Station receives a ClearDisplayMessageRequest AND Charging Station is not able to execute the request	The Charging Station SHALL respond with a ClearDisplayMessageResponse message with status: `Rejected`.

O06 - Replace DisplayMessage

No.	Type	Description
1	**Name**	Replace DisplayMessage
2	**ID**	O06
3	**Objective(s)**	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	**1.** The CSO asks the CSMS to replace an existing DisplayMessage. **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. **3.** The Charging Station accepts the request by sending a SetDisplayMessageResponse message to the CSMS. **4.** The Charging Station shows the updated/replaced message on the display at the configured moment.
	Alternative scenario(s)	O01 - Set DisplayMessage and O02 - 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.
7	**Error Handling**	n/a
8	**Remarks**	n/a

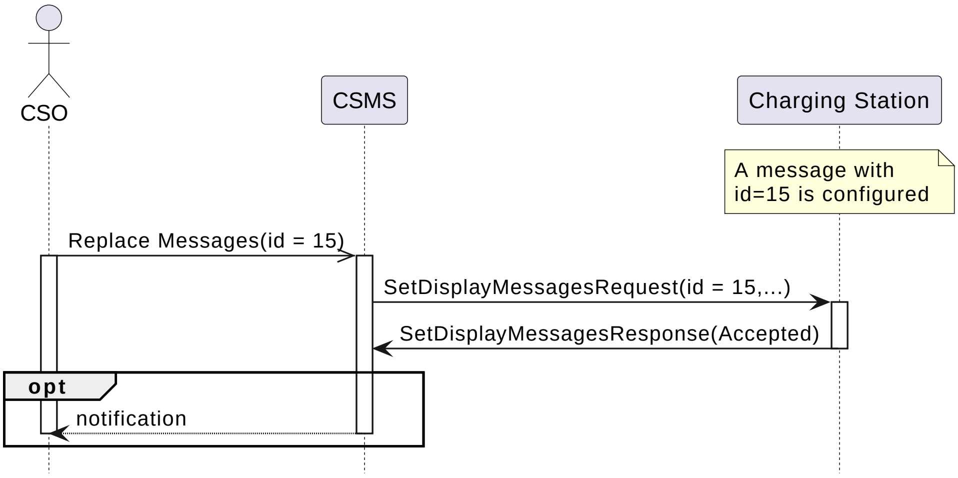

CSO
CSMS
Charging Station
A message with
id=15 is configured
Replace Messages(id = 15)
SetDisplayMessagesRequest(id = 15,...)
SetDisplayMessagesResponse(Accepted)
opt
notification

Figure 180. Replace DisplayMessage sequence diagram

O06 - Replace DisplayMessage - Requirements

Table 149. O06 - Requirements

ID	Precondition	Requirement definition
O06.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

Chapter 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 [OCPP2.1-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.

Chapter 2. Use cases & Requirements

P01 - Data Transfer to the Charging Station

No.	Type	Description
1	**Name**	Data Transfer to the Charging Station
2	**ID**	P01
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	**1.** The CSMS sends information to a Charging Station for a function not supported by OCPP with DataTransferRequest. **2.** The Charging Station responds to the CSMS with DataTransferResponse.
5	**Prerequisite(s)**	n/a
6	**Postcondition(s)**	**Successful postcondition:** DataTransferRequest is received *Successfully* and *Accepted* **Failure postcondition:** Message has been *Accepted* but the contained request is *Rejected*. In all other cases the usage of status *Accepted* or *Rejected* and the data element is part of the vendor-specific agreement between the parties involved.
7	**Error handling**	n/a
8	**Remark(s)**	Data Transfer is used if information for a function is not supported by OCPP. 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.

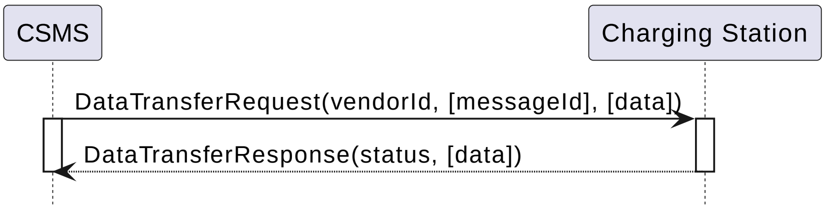

CSMS
Charging Station
DataTransferRequest(vendorId, [messageId], [data])
DataTransferResponse(status, [data])

Figure 181. Sequence Diagram: Data Transfer to the Charging Station

P01 - Data Transfer to the Charging Station - Requirements

Table 150. 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.
P01.FR.05	If the recipient of the request has no implementation for the specific vendorId.	The recipient SHALL return a status *UnknownVendor*.

ID	Precondition	Requirement definition
P01.FR.06	Upon receipt of DataTransferRequest and in case of a messageId mismatch (if used).	The recipient SHALL return status *UnknownMessageId*.
P01.FR.07		The usage of status *Accepted* or *Rejected* and the data element SHALL be part of the vendor-specific agreement between the parties involved.

P02 - Data Transfer to the CSMS

No.	Type	Description
1	**Name**	Data Transfer to the CSMS
2	**ID**	P02
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	**1.** The Charging Station sends information to the CSMS for a function not supported by OCPP with DataTransferRequest. **2.** The CSMS responds to the Charging Station with DataTransferResponse.
5	**Prerequisite(s)**	n/a
6	**Postcondition(s)**	**Successful postcondition:** DataTransferRequest is received *Successfully* and *Accepted* **Failure postcondition:** Message has been accepted but the contained request is *Rejected*. In all other cases the usage of status *Accepted* or *Rejected* and the data element is part of the vendor-specific agreement between the parties involved.
7	**Error handling**	n/a
8	**Remark(s)**	Data Transfer is used if information for a function is *not* supported by OCPP. The length of data in both the request and response message is undefined and should be agreed upon by all parties involved.

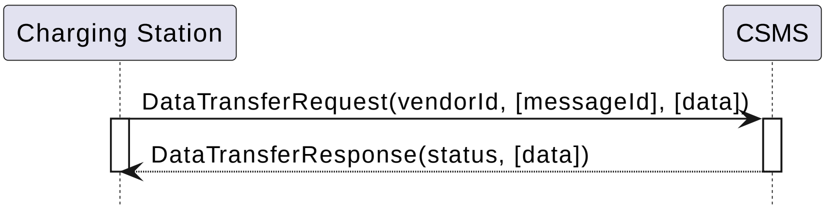

Charging Station
CSMS
DataTransferRequest(vendorId, [messageId], [data])
DataTransferResponse(status, [data])

Figure 182. Sequence Diagram: Data Transfer to the CSMS

P02 - Data Transfer to the CSMS - Requirements

Table 151. 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. (Same as P01.FR.01)
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. (Same as P01.FR.02)
P02.FR.04		The messageId in the request message MAY be used to indicate a specific message or implementation. (Same as P01.FR.03)
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. (Same as P01.FR.04)
P02.FR.06	If the recipient of the request has no implementation for the specific vendorId.	The recipient SHALL return a status *UnknownVendor*.

ID	Precondition	Requirement definition
P02.FR.07	Upon receipt of DataTransferRequest and in case of a messageId mismatch (if used).	The recipient SHALL return status *UnknownMessageId*.
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. (Same as P01.FR.07)

Q. Bidirectional Power Transfer

Chapter 1. Introduction

This document describes the communication between charging station and CSMS to allow support for bidirectional charging when either CHAdeMO or ISO 15118-20 is used for the communication between the EV and charging station.

This document uses the abbreviation V2X (vehicle-to-anything) for bidirectional power transfer between an EV and the grid (V2G), a home (V2H), a building (V2B) or an appliance (V2L). In the context of OCPP V2L can be ignored, because that does not involve any OCPP communication. V2H and V2B are specialized cases of V2G that are about using the EV battery to power a home or building, either to limit or prevent power consumption from the grid, or the provide power in case of a grid outage.

The remainder of this document does not use the abbreviation V2G for vehicle-to-grid or bidirectional power transfer to avoid any confusion with the same term that is used in ISO 15118, where it refers to the vehicle to charging station communication.

Chapter 2. Smart Charging Extensions for V2X

Support for bidirectional charging builds upon the smart charging functionality of OCPP, as described in Introduction. Charging schedules in a charging profile have been extended with additional attributes to control discharging operations.

2.1. Setpoint and charge/discharge limits

A schedule entry ChargingSchedulePeriodType has three fields to influence the charging or discharging. These settings are in Watts or Amperes, as defined by the *chargingRateUnit* of the ChargingScheduleType.

1. *setpoint*: the target value for either charging or discharging depending on the sign. Positive values indicate charging, while negative values indicate discharging.
2. *limit*: the maximum allowed charging limit. Positive values only.
3. *dischargeLimit*: the maximum allowed discharging limit. Negative values only.

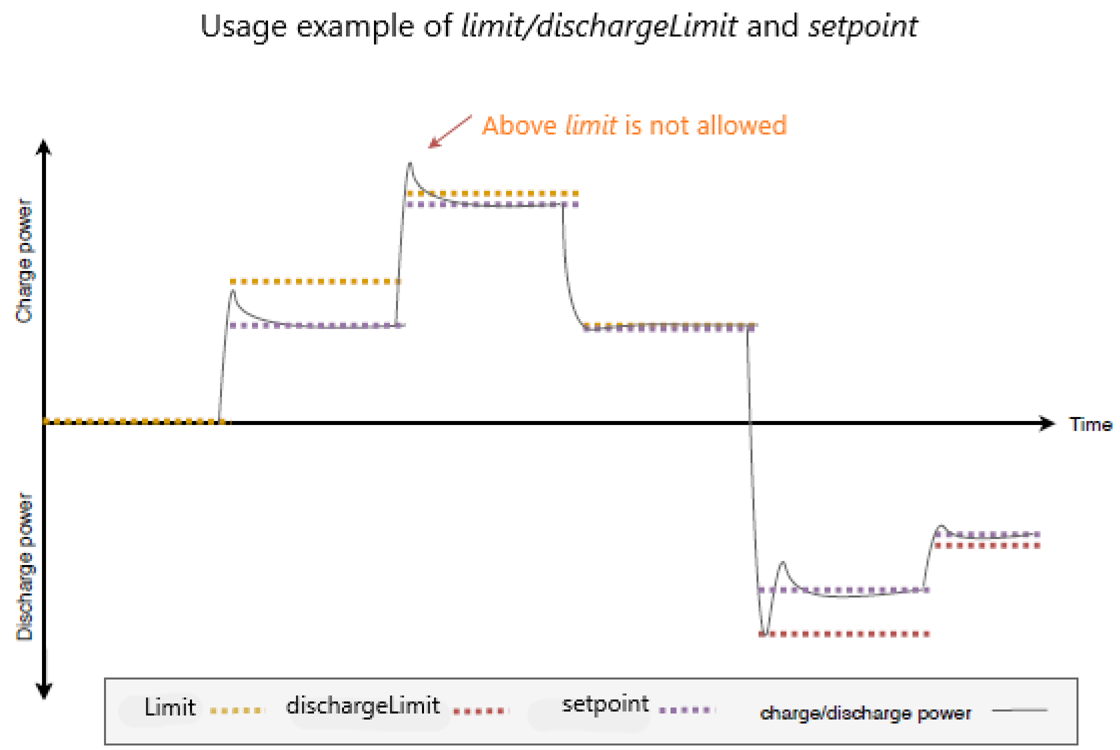

Usage example of *limit/dischargeLimit* and *setpoint*
Above *limit* is not allowed
Charge power
Discharge power
Time
Limit
dischargeLimit
setpoint
charge/discharge power

Figure 183. Example of using setpoint and limit/dischargingLimit

2.2. V2X operation modes

The various V2X use cases support different kind of operations. For example, the charging profile can define a setpoint to follow that defines the level of charging or discharging. In other situations there can be an external system, like an EMS, that defines the setpoint, or the charging profile can instruct the charging station to use local data, like a power-frequency table or load-balancing thresholds, to control the setpoint.

This mode of operation is defined by the variable *OperationMode* that is part of a ChargingSchedulePeriodType

The following operation modes exist, of which the first four are also used for regular smart charging, as described in Introduction.

1. **ChargingOnly**
2. **ExternalSetpoint**
3. **ExternalLimits**
4. **CentralSetpoint**
5. **CentralFrequency**
6. **LocalFrequency**

7. **LocalLoadBalancing**
8. **Idle**

ChargingOnly

This operation mode allows charging only, therefore is not bidirectional at all. This is also the default operation mode when the field *operationMode* is missing. It is added especially for Charging Stations that intent to operate in V2X, but are unsure during energy service negotiation if the EV and EV user are allowed to operate in V2X at this time and location. The EV then starts in this operation mode, waiting for authorization of the V2X operation mode. When V2X is authorized, a service negotiation needs to be performed between EV and charging station to start using actual V2X operation modes.

The field *limit* is used to specify the maximum allowed charging limit. The fields *setpoint* and *dischargeLimit* are not used.

ExternalSetpoint

This control mode tells the charging station that the *setpoint* parameter is to be determined by some external actor, such as an EMS. A setpoint is a charging rate that the EV must try to follow as closely as possible.

Behavior is identical to the description of ExternalSetpoint in Introduction, with the exception that *setpoint* can become negative to specify a discharging setpoint. Optionally the parameters *limit* and *dischargeLimit* can be set by CSMS to limit the range of the external setpoint.

ExternalLimits

This control mode is similar to ExternalSetpoint with the difference that it is not the *setpoint* that is controlled, but it is the *limit* and *dischargingLimit* parameters that are determined by the external actor, such as an EMS.

CentralSetpoint

In this case CSMS provides a value for *setpoint* in the charging schedule period. A *setpoint* is a charging rate that the EV must try to follow as closely as possible. Positive values for setpoint represent charging, negative values represent discharging.

If the *chargingProfileKind* = `Dynamic`, then the value of *setpoint* can be changed by CSMS during the charging schedule period via an UpdateDynamicScheduleRequest without having to submit a new charging profile.

The parameters *limit* and *dischargeLimit* are not used.

LocalFrequency

LocalFrequency is the operation mode during which the EV takes part in frequency containment reserve (FCR) services or automatic Frequency Restoration Reserve (aFRR).

FCR
FCR is a type of balancing service that is used to maintain the stability and reliability of the electricity grid. Also known as primary control reserve, it is activated when the frequency of the grid deviates from its normal operating range. When frequency deviations occur, for example, as a consequence of a power plant outage, FCR intervenes automatically within seconds in the entire synchronous area to restore the balance between supply and demand.

The power setpoint for frequency support is determined from a power/frequency (Freq-Watt) curve, based on the locally measured frequency. This operation mode requires, that the *chargingRateUnit* of the ChargingScheduleType is `W`, so that *setpoint* refers to power. The power/frequency curve might be different for various locations and is therefore provided as a field in ChargingSchedulePeriodType: *v2xFreqWattCurve*, which contains a list of at least two V2XFreqWattPointType coordinates.

Values for power in between the given frequency levels are to be determined locally by linear interpolation.

The power setpoint is depending on the frequency reading in the charging station:

```
setpoint = get power from P/F table (Frequency)
```

Example P/F table:

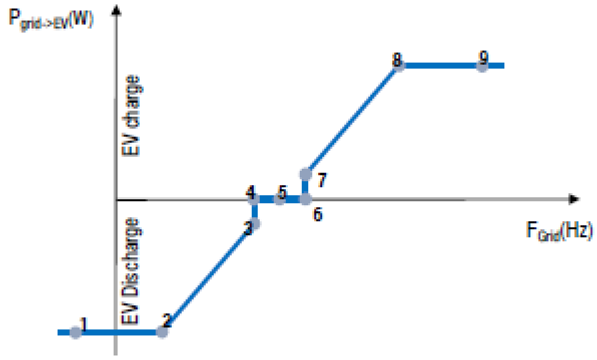


Figure 184. Example P/F table

```
# example FCR table
"v2xFreqWattCurve": [{47,-10000}, {49.80,-10000}, {49.99,-500}, {49.991,0},
{50,0}, {50.009,0}, {50.01,500}, {50.20,10000}, {53,10000}]
```

aFRR

The second form of balancing energy is automatic Frequency Restoration Reserve (aFRR). As the secondary reserve, aFRR gradually replaces FCR after 30 seconds. Once activated by the grid operator, the power must be fully available within 15 minutes, with a certain minimum ramp rate (e.g. 20%) of the offered power/min, and be able to follow a new setpoint every few seconds. It is therefore the most demanding balancing reserve product. Operators with sufficiently large energy generation or consumption can offer both upward and downward aFRR.

The aFRR capacity is activated by a signal from the TSO to the CSO. The signal is passed on to affected charging stations via the AFRRSignalRequest message. The amount of upward or downward aFRR for a charging station is set in the *v2xSignalWattCurve* element in the ChargingSchedulePeriodType.

```
# example aFRR table
"v2xSignalWattCurve": [{-1,-2000}, {0, 0}, {1,2000}]
```

CentralFrequency

CentralFrequency is similar to LocalFrequency, only in this case the setpoint for frequency support is determined by the CSMS. This is used, for instance, when costly calibrated frequency measurements are to be used that can not be installed in each charging station. This needs to be used in a charging profile with *chargingProfileKind* = `Dynamic`, such that CSMS can continually update the setpoint when the frequency changes via the UpdateDynamicScheduleRequest.

When the charging station does not receive a new *setpoint* within time, then the charging profile terminates, allowing the charging station to fall back to a charging profile with a lower stack level with a different operation mode.

LocalLoadBalancing

This operation mode allows an EV to be utilized for load balancing, for example for a building that both consumes energy and produces energy from solar panels.

Because the charging station can read the smart energy meter of the building, the charging station and EV setup can influence the resulting load on the grid connection.

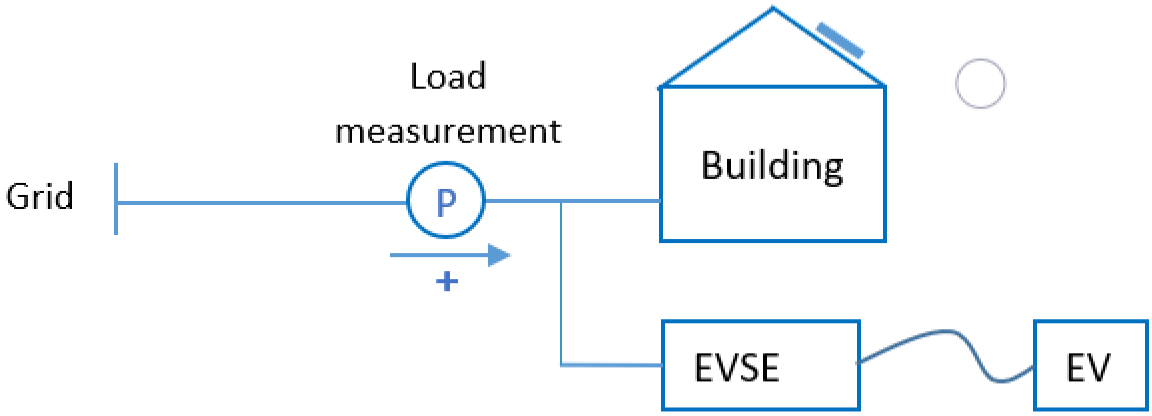


Figure 185. Diagram of a local load-balancing setup

For this operation mode there are four configurable keys:

1. `UpperThreshold`, threshold level for the measured load at which the EV will start to compensate the power of the load in order to avoid exceeding the threshold (by discharging).
2. `LowerThreshold`, threshold level for the measured load at which the EV will start to compensate the power of the load in order to avoid passing below the threshold (by charging) .
3. `UpperOffset`, the amount by which the load is compensated more (or less) than the upper threshold.
4. `LowerOffset`, the amount by which the load is compensated more (or less) than the lower threshold.

All these keys can have either positive or negative values, as long as UpperThreshold > LowerThreshold. The values of LowerOffset and UpperOffset are relative to LowerThreshold and UpperThreshold.

To better understand what possible usages of this operation mode can be, here are some examples:

1. When both LowerOffset and UpperOffset are set to zero, then the schedule will aim to keep the load exactly between LowerThreshold and UpperThreshold.
2. When UpperThreshold = 10 kW with UpperOffset = +1 kW and LowerThreshold = -5 kW with LowerOffset = -1 kW, then the schedule will aim to keep the load between +11 kW and - 6 kW.

```
Load = get latest load power reading ()

If Load > UpperThreshold
    DeltaSetpoint = UpperThreshold + UpperOffset - Load
Else If Load < LowerThreshold
    DeltaSetpoint = LowerThreshold + LowerOffset - Load
Else
    DeltaSetpoint = 0
setpoint = setpoint + DeltaSetpoint
```

The determined *setpoint* in the V2X schedule entry is always to be limited by *limit* (when positive) or *dischargeLimit* (when negative).

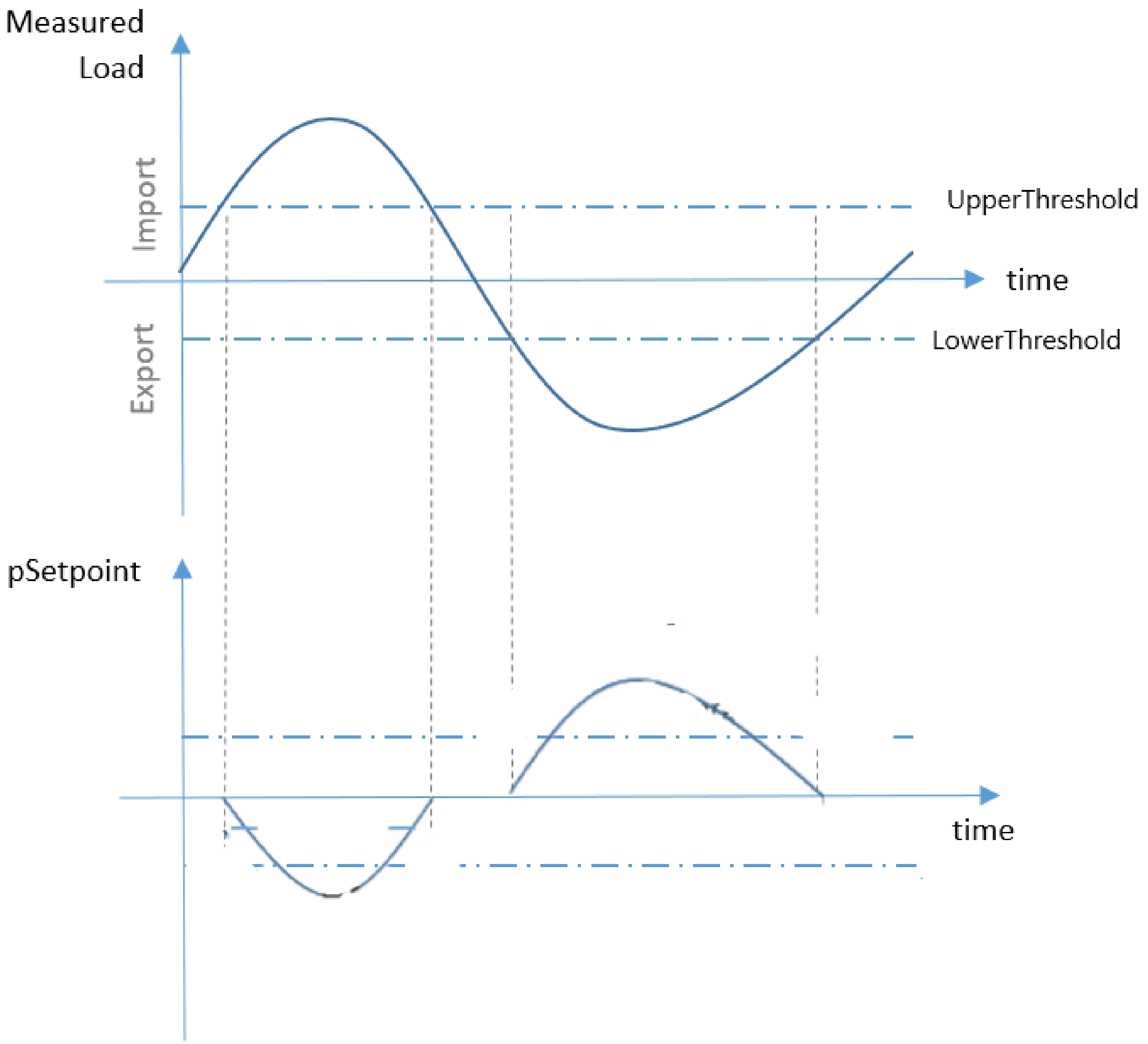


Figure 186. Graphs of measured load and compensating setpoint

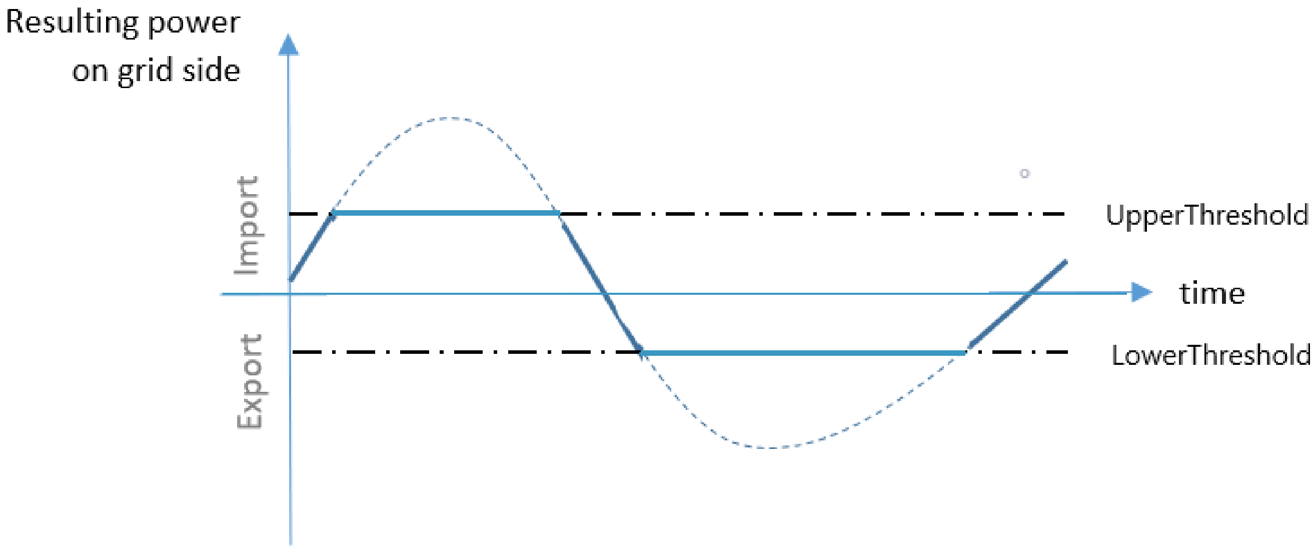


Figure 187. Graph of resulting power import/export

Two configurable threshold values `V2XLocalLoadBalancing[UpperThreshold]` and `V2XLocalLoadBalancing[LowerThreshold]` are used so the capacity controller in the charging station can limit the power imported from the grid and/or exported to the grid.

Idle

This operation mode is used when the EV is to be idle, in order to minimize the energy used by the EV for a period of time.

The field *preconditioningRequest* is used to indicate if the EV can go to sleep completely, or should be on standby. The difference between sleeping and standby is that when sleeping, the EV is not expected to react quickly to a new schedule, while for standby it is expected that the EV can quickly start charging or discharging, when for example a new schedule is sent from the CSMS to the charging station.

Limits and setpoints per operation mode

The following table lists the charging limits and setpoints that can be set for each charging mode.

limits and setpoints per operation mode	**limit**	**dischargeLimit**	**setpoint**	**setpoint-Reactive**
ChargingOnly	required	-	-	-
CentralSetpoint	optional	optional	required	optional
CentralFrequency	optional	optional	required	-
LocalFrequency	-	-	calculated from freq/signal curves	-
ExternalSetpoint	optional	optional	set externally	-
ExternalLimits	set externally	set externally	-	-
LocalLoadBalancing	-	-	calculated from load and thresholds/offsets	-

2.3. Switching V2X operation modes

The V2X operation mode is defined per ChargingSchedulePeriodType. This means that you can have a different V2X operation mode per period in a charging profile. Although this is not a likely scenario, there may be situations where it is known in advance that certain hours of the day are "ChargingOnly" and other are "LocalLoadBalancing" or "CentralSetpoint".

In many cases, though, a change of V2X operation mode is based on circumstances and cannot be planned. In that case the charging profile will have to be replaced with a new charging profile that contains a new charging schedule with the desired V2X operation mode.

It is not uncommon that a charging schedule contains only one charging schedule period, because the setpoint or limits are determined dynamically during the period. This is the case, for example, with LocalFrequency, ExternalSetpoint, ExternalLimits and LocalLoadBalancing.

2.4. Residual measurand

With the new measurand `Power.Active.Residual` the CSMS can easily monitor if the measured power is within margin of the power setpoint. When the residual is positive it is charging or discharging at a higher power than the given power setpoint, when negative it is charging or discharging at a lower power than the given power setpoint.

The power residuals are calculated by:

```
If Power.Active.Setpoint is None:
    Power.Active.Residual = 0
If Power.Active.Setpoint == 0:
    Power.Active.Residual = Power.Active.Import + Power.Active.Export
If Power.Active.Setpoint > 0:
    Power.Active.Residual = Power.Active.Import - Power.Active.Setpoint
If Power.Active.Setpoint < 0:
    Power.Active.Residual = Power.Active.Export + Power.Active.Setpoint
```

2.5. Generic smart charging rules for V2X

ID	Precondition	Requirement definition	Note
V2X.01		All requirements from section Introduction also apply to V2X	
V2X.02		A *setpoint* (or *setpointReactive*) in a charging profile is a value that EV is supposed to follow as closely as possible.	In contrast to a *limit* which only specifies the maximum power that is allowed.
V2X.03	If a ChargingSchedulePeriodType of the composite schedule contains a positive *setpoint* as well as a *limit*	Charging Station SHALL cap (i.e. not exceed) the setpoint for power towards EV to the value of *limit*.	E.g. if *setpoint* = 2000 W and *limit* = 1000 W, then the setpoint communicated to EV is 1000 W. When *chargingRateUnit* = A then *limit* is about current instead of power.
V2X.04	If a ChargingSchedulePeriodType of the composite schedule contains a negative *setpoint* as well as a *dischargingLimit*	Charging Station SHALL cap (i.e. not go below) the setpoint for discharging power towards EV to the value of *dischargingLimit*.	E.g. if *setpoint* = -2000 W and *dischargingLimit* = -1000 W, then the setpoint communicated to EV is -1000 W. This does not apply to *setpointReactive*. When *chargingRateUnit* = A then *limit* is about current instead of power.
V2X.05	Upon receiving a SetChargingProfileRequest with a ChargingSchedulePeriodType in which the value of *setpoint* lies outside the range from *dischargingLimit* to *limit*	Charging Station SHALL respond with SetChargingProfileResponse with *status* = `Rejected` and *reasonCode* = `InvalidSchedule`	E.g. Reject if *setpoint* = -750 W when *limit* = 1000 W and *dischargingLimit* = -500 W.

Chapter 3. Use Cases & Requirements

Q01 - V2X Authorization

New in OCPP 2.1

No.	Type	Description
1	**Name**	V2X Authorization
2	**ID**	Q01
3	**Objective(s)**	Authorization of an EV by the CSMS to start a V2X power transfer loop.
4	**Description**	Upon authorization the EV receives a list of allowed energy transfers from CSMS. The EV will select the desired energy transfer from that list and report it to CSMS.
	Actors	EV, Charging Station, CSMS (possibly a secondary actor such as an aggregating party)
	Scenario description	1. The Charging Station sends a AuthorizeRequest with EVCCID of EV in *additionalInfo* of *idToken*. a. The CSMS returns an AuthorizeResponse with *idTokenInfo.status* = `Accepted` and the list *allowedEnergyTransfer* for this EV. 2. The Charging Station sends a TransactionEventRequest, with *eventType* set to `Started` a. The CSMS will respond with the TransactionEventResponse. 3. The EV and Charging station agree on an energy transfer method. 4. The Charging Station sends NotifyEVChargingNeedsRequest with the chosen *requestedEnergyTransfer* (and optionally the *availableEnergyTransfer* list), together with *v2xcChargingParameters*, *departureTime* and a ControlModeEnumType. a. The CSMS will respond with the NotifyEVChargingNeedsResponse message, with *status* set to `Accepted`. 5. The CSMS will send the SetChargingProfileRequest message to the Charging Station with a charging schedule containing a V2X *operationMode* other than `ChargingOnly`. a. The Charging Station will respond with the SetChargingProfileResponse message. 6. The Charging Station will send its parameters on charging and discharging to the EV. 7. The EV will request to start V2X power transfer. a. The Charging station will accept to start V2X power transfer.
	Scenario description #2	*When CSMS decides to rely on a `TxDefaultProfile` with a charging schedule containing a V2X operationMode other than `ChargingOnly`:* Steps 1 to 3 as above. 4. The Charging Station sends NotifyEVChargingNeedsRequest with the chosen *requestedEnergyTransfer* (and optionally the *availableEnergyTransfer* list), together with *v2xcChargingParameters*, *departureTime* and a ControlModeEnumType. a. The CSMS will respond with the NotifyEVChargingNeedsResponse message, with *status* set to `NoChargingProfile`, to signal that no SetChargingProfileRequest is to be expected. 5. The Charging Station uses the `TxDefaultProfile`. 6. The Charging Station will send its parameters on charging and discharging to the EV. 7. The EV will request to start V2X power transfer. a. The Charging station will accept to start V2X power transfer.
5	**Prerequisites**	ISO15118Ctrlr.Enabled = true and V2XChargingCtrlr.Enabled = true
6	**Post conditions**	

No.	Type	Description
7	**Error Handling**	When *requestedEnergyTransfer* is NOT accepted by the CSMS: 1. The Charging Station uses the NotifyEVChargingNeedsRequest message to send the chosen *requestedEnergyTransfer* to the CSMS. 2. The CSMS will respond with the NotifyEVChargingNeedsResponse message, with *status* set to `Rejected`. 3. The Charging Station will stop the current transaction.
8	**Remarks**	A V2X energy transfer may likely require authorization from a third party (e.g. the eMSP). When Charging Station requests a V2X energy service, there is a possibility that the CSMS cannot allow that energy service within the timeout of the EV. It can then only Reject it and the transaction will stop. It is therefore advised to always start by requesting a `ChargingOnly` energy service, but indicate that the V2X service is available for this EV. Once the Charging Station receives the *allowedEnergyTransfer* including the V2X service from the CSMS, there can be a service renegotiation to change to the V2X service. The ISO 15118 EVCCID of the EV is available in the device model as the variable *VehicleId* from the component *ConnectedEV*, if that is needed by the operator to determine whether to allow V2X operations. In some cases the EVCCID may be used as an *idToken* for authorization, or it can be added to the *additionalInfo* field of the *idToken* that is used for authorization.

NOTE The ISO 15118-20 ScheduleExchangeReq has a timeout of 2s, but this can be prolonged up to 1 minute if the Charging Station returns a ScheduleExchangeRes(*EVSEProcessing* = Ongoing) response before the 2s timeout. This will trigger the EV to send another ScheduleExchangeReq() to which the Charging Station can eventually respond with ScheduleExchangeRes(*EVSEProcessing* = Finished) once the NotifyEVChargingNeedsRequest/Response and SetChargingProfileRequest/Response have been exchanged between Charging Station and CSMS.

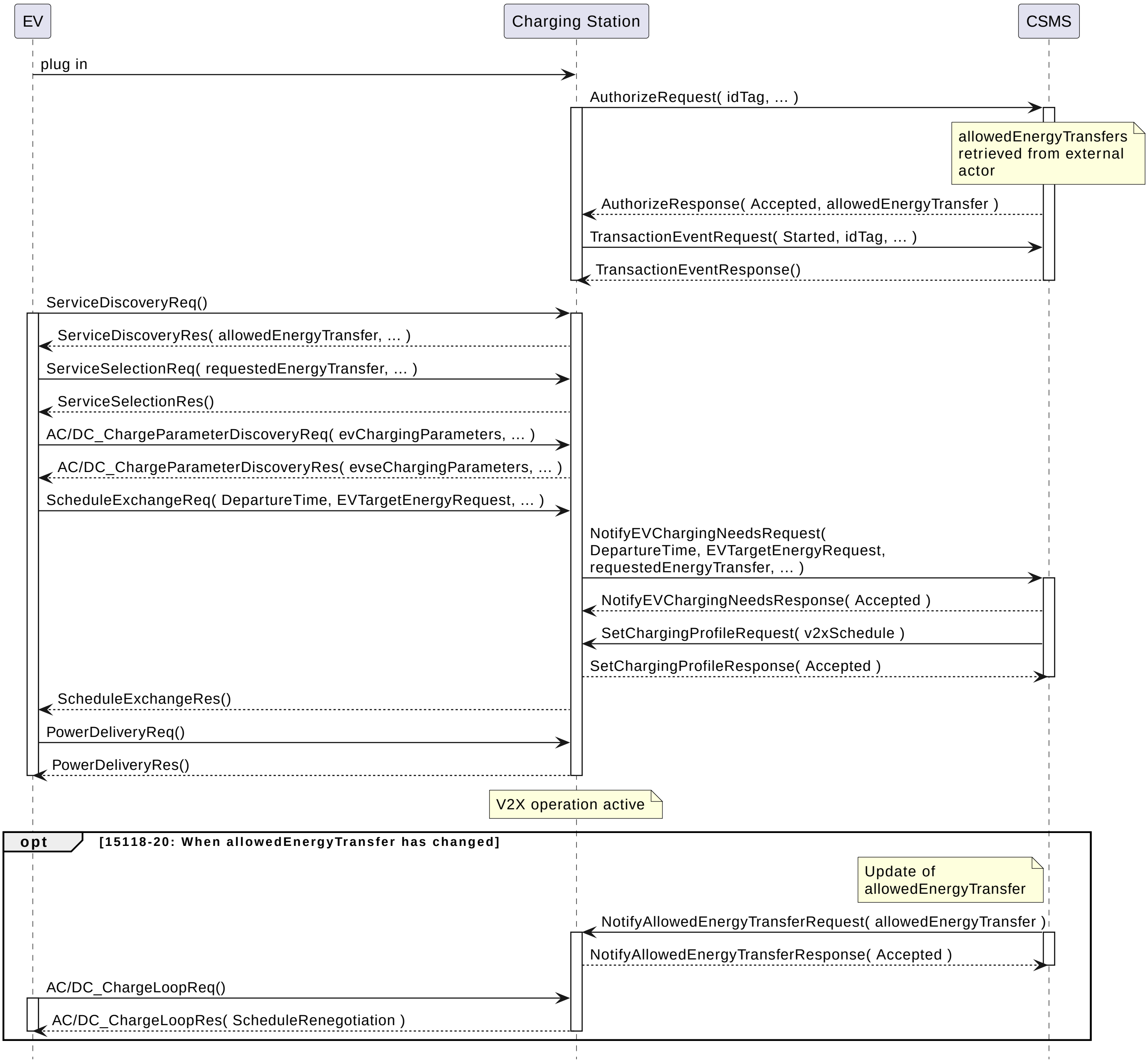

EV
Charging Station
CSMS
plug in
AuthorizeRequest(idTag, ...)
allowedEnergyTransfers retrieved from external actor
AuthorizeResponse(Accepted, allowedEnergyTransfer)
TransactionEventRequest(Started, idTag, ...)
TransactionEventResponse()
ServiceDiscoveryReq()
ServiceDiscoveryRes(allowedEnergyTransfer, ...)
ServiceSelectionReq(requestedEnergyTransfer, ...)
ServiceSelectionRes()
AC/DC_ChargeParameterDiscoveryReq(evChargingParameters, ...)
AC/DC_ChargeParameterDiscoveryRes(evseChargingParameters, ...)
ScheduleExchangeReq(DepartureTime, EVTargetEnergyRequest, ...)
NotifyEVChargingNeedsRequest(DepartureTime, EVTargetEnergyRequest, requestedEnergyTransfer, ...)
NotifyEVChargingNeedsResponse(Accepted)
SetChargingProfileRequest(v2xSchedule)
SetChargingProfileResponse(Accepted)
ScheduleExchangeRes()
PowerDeliveryReq()
PowerDeliveryRes()
V2X operation active
opt
[15118-20: When allowedEnergyTransfer has changed]
Update of allowedEnergyTransfer
NotifyAllowedEnergyTransferRequest(allowedEnergyTransfer)
NotifyAllowedEnergyTransferResponse(Accepted)
AC/DC_ChargeLoopReq()
AC/DC_ChargeLoopRes(ScheduleRenegotiation)

Figure 188. V2X Authorization

Q01 - V2X Authorization - Requirements

Table 152. Q01 - Requirements

ID.	Precondition	Requirements	Note
AllowedEnergyTransfer / RequestedEnergyTransfer			
Q01.FR.01	When Charging Station's ISO15118Ctrlr.Enabled = true and V2XChargingCtrlr.Enabled = true AND CSMS receives an AuthorizeRequest AND CSMS is able to determine a list of *allowedEnergyTransfer* before sending the AuthorizeResponse	CSMS SHALL return a list of *allowedEnergyTransfer* in AuthorizeResponse.	No need to return *allowedEnergyTransfer* for non-15118 stations or non-V2X capable stations.

ID.	Precondition	Requirements	Note
Q01.FR.02	When Charging Station starts an ISO 15118-20 transaction	Charging Station SHALL add EVCCID to *idToken* in TransactionEventRequest(*eventType*=`Started`) in *idToken.additionalInfo.additionalIdToken* and with *idToken.additionalInfo.type* set to "EVCCID".	This transaction may become bidirectional. This is needed in case CSMS uses the EVCCID of vehicle to decide whether to allow V2X.
Q01.FR.03	Q01.FR.02	Charging Station SHALL send a NotifyEVChargingNeedsRequest with *evseId* set to the EVSE used for this transaction AND the *requestedEnergyTransfer* set to the type that has been agreed between EV and Charging Station AND a list of *availableEnergyTransfer* that is supported by EV AND the fields *controlMode*, *v2xChargingParameters* and optionally *departureTime*, *mobilityNeedsMode* and *evEnergyOffer*.	
Q01.FR.04	If CSMS does not accept the *requestedEnergyTransfer*	CSMS SHALL respond with a NotifyEVChargingNeedsResponse with *status* = `Rejected`.	Charging station will then stop the transaction.
Q01.FR.05	Q01.FR.04	Charging Station SHOULD start a service renegotiation with EV for a different energy transfer service	This situation should not occur when an energy transfer is selected from the *allowedEnergyTransfer* list in the AuthorizeResponse.
Q01.FR.06	Q01.FR.04 AND Charging Station did not start a service renegotiation	Charging Station SHALL terminate the transaction and send a TransactionEventRequest with *eventType*=Ended, *triggerReason* =`AbnormalCondition`, *stoppedReason*=`ReqEnergyTransferRejected`.	This is an error situation, because energy service did not match *allowedEnergyTransfer*.
Q01.FR.07	If CSMS accepts the *requestedEnergyTransfer*	CSMS SHALL respond with a NotifyEVChargingNeedsResponse with *status* = `Accepted` or `Processing`.	Charging station can expect to receive a charging profile immediately or soon.
Q01.FR.08	Q01.FR.01 AND CSMS is not able to determine a list of *allowedEnergyTranfer* before sending the AuthorizeResponse	CSMS SHALL omit *allowedEnergyTransfer* from AuthorizeResponse.	This can happen if it could not be determined within the short time span before the response has to be returned, e.g. because a third party has to be requested for permission.
Q01.FR.09	Q01.FR.20	Charging Station SHALL send a NotifyEVChargingNeedsRequest with *evseId* set to the EVSE used for this transaction and *requestedEnergyTransfer* set to its default energy transfer (charging only AC/DC) and *availableEnergyTransfer* set to the supported energy transfers.	Depending on type of EVSE this will be AC_single_phase, AC_two_phase, AC_three_phase or DC, DC_ACDP
Device model			
Q01.FR.30		Charging Station SHALL report the operation modes that it supports in V2XChargingCtrlr SupportedOperationModes	
Q01.FR.31		Charging Station SHALL at least support the operation modes `ChargingOnly`, `CentralSetpoint` and `CentralFrequency`.	`CentralFrequency` is basically a `CentralSetpoint` for a specific use case.

ID.	Precondition	Requirements	Note
Q01.FR.32		Charging Station SHALL report the energy transfer modes that it supports in V2XChargingCtrlr SupportedEnergyTransferModes	
Q01.FR.33	When V2XChargingCtrlr TxStartedMeasurands is defined for the *instance* matching the *operationMode* of the current ChargingSchedulePeriodType	Charging Station SHALL include those measurands in *meterValues* of TransactionEventRequest in addition to measurands already defined in SampledDataCtrlr TxStartedMeasurands.	
Q01.FR.34	When V2XChargingCtrlr TxEndedMeasurands is defined for the *instance* matching the *operationMode* of the current ChargingSchedulePeriodType	Charging Station SHALL include those measurands in *meterValues* of TransactionEventRequest in addition to measurands already defined in SampledDataCtrlr TxEndedMeasurands.	
Q01.FR.35	When V2XChargingCtrlr TxUpdatedMeasurands is defined for the *instance* matching the *operationMode* of the current ChargingSchedulePeriodType	Charging Station SHALL include those measurands in *meterValues* of TransactionEventRequest in addition to measurands already defined in SampledDataCtrlr TxUpdatedMeasurands.	
Q01.FR.36		Charging Station SHALL support the device model component Connected EV related with at least the variables "Available", "VehicleId", "VehicleCertificate", "ProtocolAgreed", "ProtocolSupportedByEV".	

Q02 - Starting in operationMode ChargingOnly before enabling V2X

New in OCPP 2.1

No.	Type	Description
1	**Name**	Starting in operationMode ChargingOnly before enabling V2X
2	**ID**	Q02
3	**Objective(s)**	To start a transaction with operationMode ChargingOnly, such that it can become a bidirectional session at a later time.
4	**Description**	This operation mode is added especially for Charging Stations that intent to operate in V2X, but are unsure during energy service negotiation if the EV is allowed to operate in V2X at this time and location. Charging Station starts in this operation mode, waiting for authorization of the V2X operation mode. When V2X is authorized, a service negotiation should be performed between EV and Charging Station to start using actual V2X operation modes.
	Actors	EV, Charging Station, CSMS

No.	Type	Description
	Scenario description	1. The Charging Station sends a AuthorizeRequest with EVCCID of EV in *additionalInfo* of *idToken*. a. The CSMS returns an AuthorizeResponse with *idTokenInfo.status* = `Accepted` and the list *allowedEnergyTransfer* without V2X. 2. The Charging Station sends a NotifyEVChargingNeedsRequest with *requestedEnergyTransfer* set to normal AC or DC. 3. The CSMS will respond with the NotifyEVChargingNeedsResponse message, with *status* set to `Accepted` 4. The CSMS sends a SetChargingProfileRequest message, with a *chargingSchedule* with *operationMode* set to `ChargingOnly`. a. Charging Station and CSMS behave as described in Use cases & Requirements. 5. The CSMS gets the EVCCID from EV from *additionalInfo* which it had received in the AuthorizeRequest or from the device model variable ConnectedEV.VehicleId. 6. Alternatively, CSMS gets the vehicle certificate from the device model variable ConnectedEV.VehicleCertificate. 7. When CSMS decides to allow V2X, for example based on *idToken*, EVCCID or vehicle certificate a. CSMS sends a NotifyAllowedEnergyTransferRequest to Charging Station with an updated *allowedEnergyTransfer* list that contains a bidirectional transfer, DC_BPT, AC_BPT or AC_BPT_DER. b. Charging Station responds with NotifyAllowedEnergyTransferResponse with *status* = `Accepted`. 8. The Charging Station will start a service renegotiation with the EV, allowing the EV to change its current energy service to one with potentially more value.
5	**Prerequisites**	
6	**Post conditions**	Charging profile schedule ends
7	**Error Handling**	
8	**Remarks**	This use case may also be applied when the user authorizes using external identification means (EIM) before plugging-in a cable. In that case, the charging station does not know which EVSE will be selected by the user (in case multiple EVSEs are available). If the EVSEs have different capabilities (e.g. AC and DC), then the CSMS will not be able to provide the correct list of allowed energy transfer options in the AuthorizationResponse at time of authorization. When the user connects the cable to EV, then CSMS can send a NotifyAllowedEnergyTransferRequest to update the list of allowed energy transfers for this EVSE. This will trigger a service renegotiation, as described in this use case.

NOTE The diagram below only shows ISO 15118-20 messages for as far as they are relevant for the data exchange between Charging Stations and CSMS.

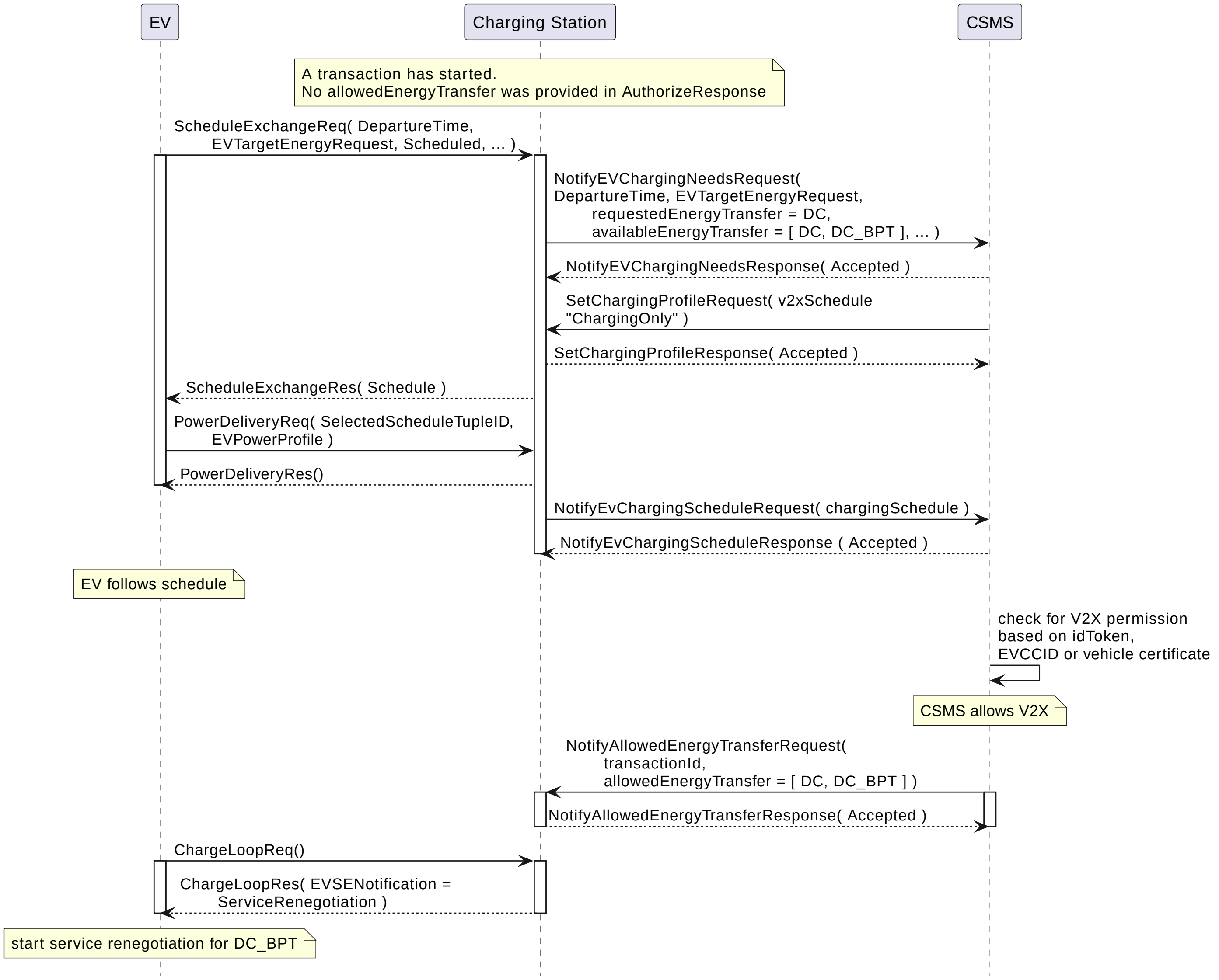

EV
Charging Station
CSMS
A transaction has started.
No allowedEnergyTransfer was provided in AuthorizeResponse
ScheduleExchangeReq(DepartureTime, EVTargetEnergyRequest, Scheduled, ...)
NotifyEVChargingNeedsRequest(DepartureTime, EVTargetEnergyRequest, requestedEnergyTransfer = DC, availableEnergyTransfer = [DC, DC_BPT], ...)
NotifyEVChargingNeedsResponse(Accepted)
SetChargingProfileRequest(v2xSchedule "ChargingOnly")
SetChargingProfileResponse(Accepted)
ScheduleExchangeRes(Schedule)
PowerDeliveryReq(SelectedScheduleTupleID, EVPowerProfile)
PowerDeliveryRes()
NotifyEvChargingScheduleRequest(chargingSchedule)
NotifyEvChargingScheduleResponse (Accepted)
EV follows schedule
check for V2X permission based on idToken, EVCCID or vehicle certificate
CSMS allows V2X
NotifyAllowedEnergyTransferRequest(transactionId, allowedEnergyTransfer = [DC, DC_BPT])
NotifyAllowedEnergyTransferResponse(Accepted)
ChargeLoopReq()
ChargeLoopRes(EVSENotification = ServiceRenegotiation)
start service renegotiation for DC_BPT

Figure 189. V2X Operation Mode ChargingOnly before enabling V2X

Q02 - Charging only (V2X control) before starting V2X - Requirements

Table 153. Q02 - Requirements

ID.	Precondition	Requirements	Note
Q02.FR.01	When Charging Station does not request bidirectional power transfer in NotifyEVChargingNeedsRequest	CSMS SHALL send a SetChargingProfileRequest message, with *chargingSchedulePeriod(s)* with field *operationMode* set to `ChargingOnly` or omitted	When *operationMode* is omitted, this also defaults to `ChargingOnly`.
Q02.FR.02	Q02.FR.01	CSMS SHALL NOT include fields *dischargeLimit*, *setpoint*, *setpointReactive* in the ChargingProfileType.	This also includes the L2 and L3 variants of those fields.
Q02.FR.03	When CSMS is not able to determine a list of *allowedEnergyTransfer* before sending the AuthorizeResponse	CSMS SHALL omit *allowedEnergyTransfer* from AuthorizeResponse.	This can happen if it could not be determined within the short time span before the response has to be returned, e.g. because a third party has to be requested for permission. (Same as Q01.FR.08)

ID.	Precondition	Requirements	Note
Q02.FR.04	Q02.FR.03	Charging Station SHALL send a NotifyEVChargingNeedsRequest with *evseId* set to the EVSE used for this transaction and *requestedEnergyTransfer* set to its default energy transfer (charging only AC/DC) and *availableEnergyTransfer* set to the supported energy transfers.	Depending on type of EVSE this will be AC_single_phase, AC_two_phase, AC_three_phase or DC, DC_ACDP
Q02.FR.05	When CSMS changes the list of allowed energy transfers for a transaction	CSMS SHALL send a NotifyAllowedEnergyTransferRequest with the *allowedEnergyTransfer* list and with *transactionId* set to the transaction for which this is intended.	For example, when permission for V2X is received some time after authorization.
Q02.FR.06	Q02.FR.05	Charging Station SHALL respond with a NotifyAllowedEnergyTransferResponse with *status* = `Accepted` and start an ISO 15118 service renegotiation with EV	
Q02.FR.07	Q02.FR.06	Charging Station SHALL send a NotifyEVChargingNeedsRequest as in Q01.FR.03 with an updated list of *requestedEnergyTransfer*.	

Q03 - Central V2X control with charging schedule

New in OCPP 2.1

No.	Type	Description
1	**Name**	Central V2X control with charging schedule
2	**ID**	Q03
3	**Objective(s)**	To allow the CSMS to control the charge and discharge behaviour of an EV with power profiles.
4	**Description**	The EV's charging and discharging behaviour is controlled by the Central System. The CSMS sends a power profile and can update it at any time. The power profile might be determined by some secondary actor that relays through the CSMS.
	Actors	EV, Charging Station, CSMS, [secondary actor]
	Scenario description	1. The CSMS has sent a SetChargingProfileRequest message with a *chargingProfilePurpose* `TxProfile` or `TxDefaultProfile` and with a *chargingSchedule* that has one or more periods with *operationMode* set to `CentralSetpoint` and *setpoint* to the desired active power the EV should import or export. 2. In ISO 15118 Scheduled mode the Charging Station will send the schedule to the EV to be executed. A change to the schedule will trigger a renegotiation in ISO 15118. 3. In ISO 15118 Dynamic mode the Charging Station will set the charge or discharge rate of the EV to *setpoint* at start of each schedule period. No schedule is sent to EV and no renegotiation is triggered in ISO 15118. If the charging profile is of *chargingProfileKind* = `Dynamic`, then CSMS can update the *setpoint* via the UpdateDynamicScheduleRequest message.
5	**Prerequisites**	A charging profile from CSMS must have profile purpose `TxProfile` or `TxDefaultProfile` for V2X operations.
6	**Post conditions**	Charging profile ends
7	**Error Handling**	
8	**Remarks**	CSMS can control with the fields *maxOfflineDuration* and *invalidAfterOfflineDuration* of a ChargingProfileType how long it remains valid in an offline situation, and whether it can be resumed once online again.

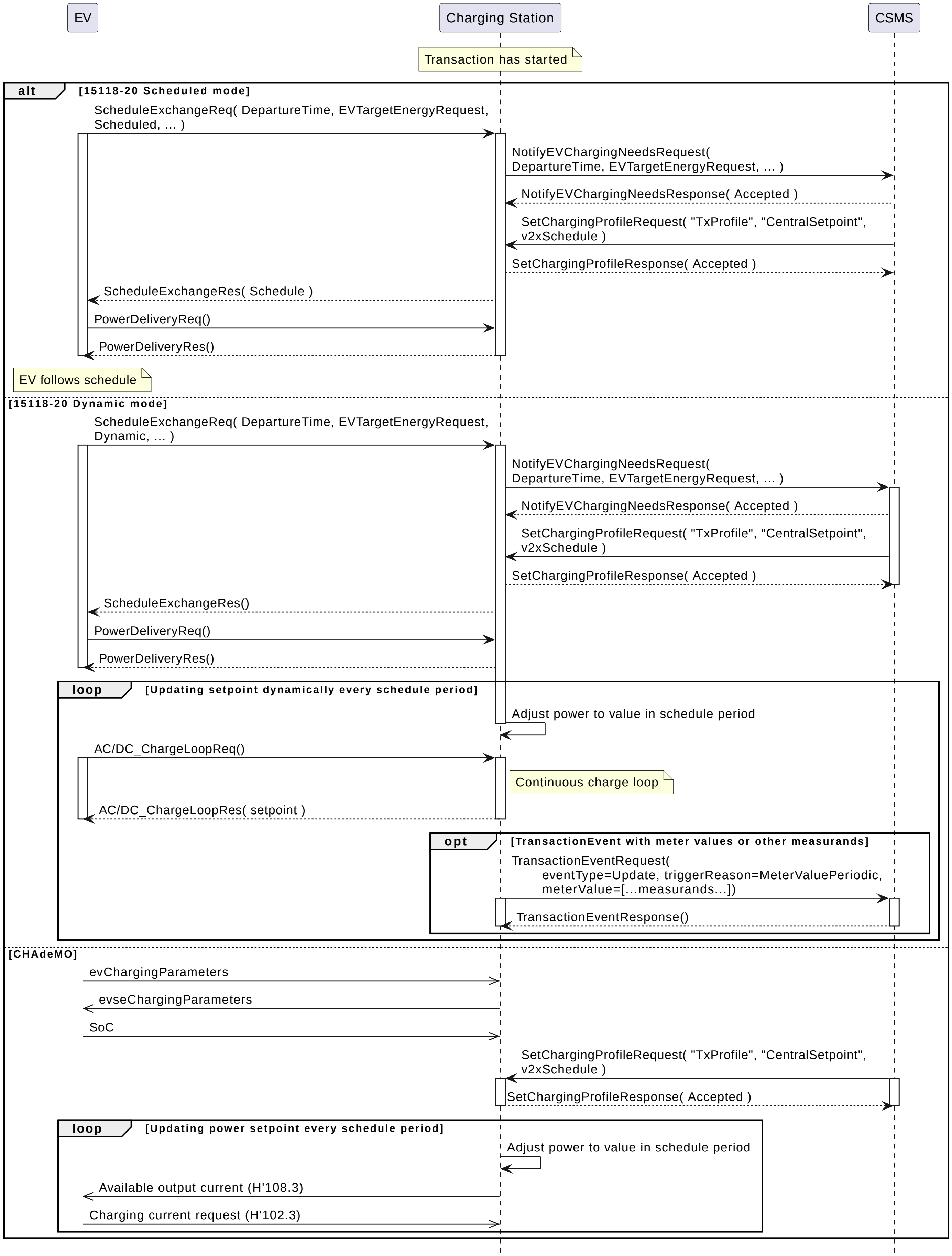

EV
Charging Station
CSMS
Transaction has started
alt
[15118-20 Scheduled mode]
ScheduleExchangeReq(DepartureTime, EVTargetEnergyRequest, Scheduled, ...)
NotifyEVChargingNeedsRequest(DepartureTime, EVTargetEnergyRequest, ...)
NotifyEVChargingNeedsResponse(Accepted)
SetChargingProfileRequest("TxProfile", "CentralSetpoint", v2xSchedule)
SetChargingProfileResponse(Accepted)
ScheduleExchangeRes(Schedule)
PowerDeliveryReq()
PowerDeliveryRes()
EV follows schedule
[15118-20 Dynamic mode]
ScheduleExchangeReq(DepartureTime, EVTargetEnergyRequest, Dynamic, ...)
NotifyEVChargingNeedsRequest(DepartureTime, EVTargetEnergyRequest, ...)
NotifyEVChargingNeedsResponse(Accepted)
SetChargingProfileRequest("TxProfile", "CentralSetpoint", v2xSchedule)
SetChargingProfileResponse(Accepted)
ScheduleExchangeRes()
PowerDeliveryReq()
PowerDeliveryRes()
loop
[Updating setpoint dynamically every schedule period]
Adjust power to value in schedule period
AC/DC_ChargeLoopReq()
Continuous charge loop
AC/DC_ChargeLoopRes(setpoint)
opt
[TransactionEvent with meter values or other measurands]
TransactionEventRequest(eventType=Update, triggerReason=MeterValuePeriodic, meterValue=[...measurands...])
TransactionEventResponse()
[CHAdeMO]
evChargingParameters
evseChargingParameters
SoC
SetChargingProfileRequest("TxProfile", "CentralSetpoint", v2xSchedule)
SetChargingProfileResponse(Accepted)
loop
[Updating power setpoint every schedule period]
Adjust power to value in schedule period
Available output current (H'108.3)
Charging current request (H'102.3)

Figure 190. V2X Operation Mode Central Setpoint

Q03 - Central V2X control with charging schedule - Requirements

Table 154. Q03 - Requirements

ID.	Precondition	Requirements	Note
OperationMode CentralSetpoint			
Q03.FR.01	When CSMS provides setpoints for charging/discharging	CSMS SHALL send a SetChargingProfileRequest message with one or more *chargingSchedulePeriod(s)* that have *operationMode* = `CentralSetpoint`.	
Q03.FR.02	Q03.FR.01	CSMS SHALL NOT include fields *limit* and *dischargeLimit* in the ChargingSchedulePeriodType.	This also includes the L2 and L3 variants of those fields.

Q04 - Central V2X control with dynamic CSMS setpoint

New in OCPP 2.1

No.	Type	Description
1	**Name**	Central V2X control with dynamic CSMS setpoint
2	**ID**	Q04
3	**Objective(s)**	To allow the CSMS to control the charge and discharge behaviour of an EV by dynamically changing a setpoint (instead of providing a schedule).
4	**Description**	The EV's charging and discharging behaviour is controlled by the Central System. The CSMS sends a dynamic charging profile with a single period, which is updated frequently.
	Actors	EV, Charging Station, CSMS, [secondary actor]
	Scenario description	CSMS periodically updates the charging schedule: 1. The CSMS has sent a SetChargingProfileRequest message with a *chargingProfilePurpose* `TxProfile` or `TxDefaultProfile` and a *chargingProfileKind* = `Dynamic`. The *chargingProfile* contains 1 *chargingSchedule* with 1 *chargingSchedulePeriod* which has *operationMode* set to `CentralSetpoint` and *setpoint* to the desired active power the EV should import or export. 2. If *chargingSchedule.duration* has a value, then this specifies how long the *chargingSchedule* remains valid after the SetChargingProfileRequest or after the last update by UpdateDynamicScheduleRequest. 3. The Charging Station will obey the values of the (one and only) *chargingSchedulePeriod*, e.g. it will set the charge or discharge rate of the EV to *setpoint*. 4. CSMS updates the *chargingSchedulePeriod* periodically by sending a UpdateDynamicScheduleRequest. If *chargingSchedule.duration* is set and the *chargingSchedulePeriod* is not updated after *duration* seconds, then the *chargingSchedule* ends and Charging Station will fall back to the next valid charging profile.

No.	Type	Description
	Scenario description #2	Charging Stations pulls charging schedule updates from CSMS: 1. The CSMS has sent a SetChargingProfileRequest message with a *chargingProfilePurpose* `TxProfile` or `TxDefaultProfile` and a *chargingProfileKind* = `Dynamic` and a *dynUpdateInterval* = 60. The *chargingProfile* contains 1 *chargingSchedule* with 1 *chargingSchedulePeriod* which has *operationMode* set to `CentralSetpoint` and *setpoint* to the desired active power the EV should import or export. 2. If *chargingSchedule.duration* has a value, then this specifies how long the *chargingSchedule* remains valid after SetChargingProfileRequest or UpdateDynamicScheduleRequest. 3. The Charging Station will obey the values of the (one and only) *chargingSchedulePeriod*, e.g. it will set the charge or discharge rate of the EV to *setpoint*. 4. At *dynUpdateInterval* seconds after last update of the charging profile, Charging Station sends a PullDynamicScheduleUpdateRequest to CSMS. 5. CSMS responds with PullDynamicScheduleUpdateResponse with new values to use in ChargingSchedulePeriodType. If *chargingSchedule.duration* is set and the *chargingSchedulePeriod* is not updated after *duration* seconds, then the *chargingSchedule* ends and Charging Station will fall back to the next valid charging profile.
5	**Prerequisites**	Profile purpose must be `TxProfile` or `TxDefaultProfile` and profile kind must be `Dynamic`.
6	**Post conditions**	Charging profile ends, because no update is received after *duration* seconds or charging profile is no longer the active profile.
7	**Error Handling**	
8	**Remarks**	A dynamic charging profile does not necessarily require the use of ISO 15118-20 dynamic control mode. Example of changing a dynamic setpoint to 2000

```
"UpdateDynamicScheduleRequest":
{
  "chargingProfileId": 10,
  "setpoint": 2000
}
```

or

```
"PullDynamicScheduleUpdateResponse":
{
  "setpoint": 2000
}
```

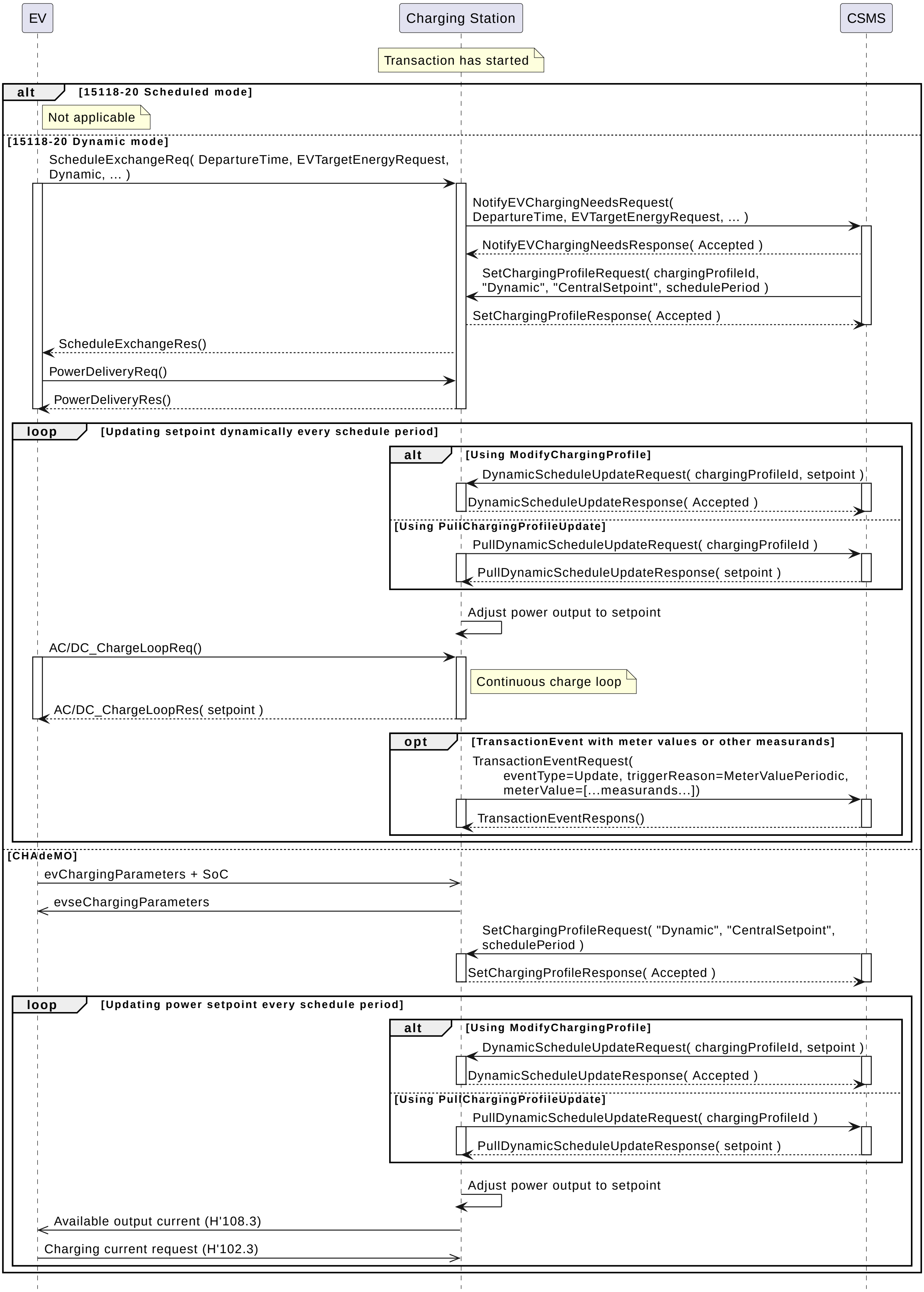

EV
Charging Station
CSMS
Transaction has started
alt [15118-20 Scheduled mode]
Not applicable
[15118-20 Dynamic mode]
ScheduleExchangeReq(DepartureTime, EVTargetEnergyRequest, Dynamic, ...)
NotifyEVChargingNeedsRequest(DepartureTime, EVTargetEnergyRequest, ...)
NotifyEVChargingNeedsResponse(Accepted)
SetChargingProfileRequest(chargingProfileId, "Dynamic", "CentralSetpoint", schedulePeriod)
SetChargingProfileResponse(Accepted)
ScheduleExchangeRes()
PowerDeliveryReq()
PowerDeliveryRes()
loop [Updating setpoint dynamically every schedule period]
alt [Using ModifyChargingProfile]
DynamicScheduleUpdateRequest(chargingProfileId, setpoint)
DynamicScheduleUpdateResponse(Accepted)
[Using PullChargingProfileUpdate]
PullDynamicScheduleUpdateRequest(chargingProfileId)
PullDynamicScheduleUpdateResponse(setpoint)
Adjust power output to setpoint
AC/DC_ChargeLoopReq()
Continuous charge loop
AC/DC_ChargeLoopRes(setpoint)
opt [TransactionEvent with meter values or other measurands]
TransactionEventRequest(eventType=Update, triggerReason=MeterValuePeriodic, meterValue=[...measurands...])
TransactionEventRespons()
[CHAdeMO]
evChargingParameters + SoC
evseChargingParameters
SetChargingProfileRequest("Dynamic", "CentralSetpoint", schedulePeriod)
SetChargingProfileResponse(Accepted)
loop [Updating power setpoint every schedule period]
alt [Using ModifyChargingProfile]
DynamicScheduleUpdateRequest(chargingProfileId, setpoint)
DynamicScheduleUpdateResponse(Accepted)
[Using PullChargingProfileUpdate]
PullDynamicScheduleUpdateRequest(chargingProfileId)
PullDynamicScheduleUpdateResponse(setpoint)
Adjust power output to setpoint
Available output current (H'108.3)
Charging current request (H'102.3)

Figure 191. V2X Operation Mode with CentralSetpoint set dynamically

NOTE This use case adheres to requirements related to CentralSetpoint from Q03 and Dynamic charging profiles from

> K01. There are no specific requirements for this use case.

Q05 - External V2X setpoint control with a charging profile from CSMS

New in OCPP 2.1

No.	Type	Description
1	**Name**	External V2X setpoint control with a charging profile from CSMS
2	**ID**	Q05
3	**Objective(s)**	CSMS explicitly allows an External System to control the charge and discharge behaviour of an EV for a certain period of time.
4	**Description**	CSMS has disallowed that an external actor submits a `ChargingStationExternalConstraints` charging profile. Instead, CSMS defines a charging profile with a *chargingSchedulePeriod* that has *operationMode* = `ExternalLimits` or `ExternalSetpoint`. Only during this charging period CSMS lets the charging and discharging behaviour be controlled by an External System connected to the Charging Station, such as an EMS.
	Actors	EV, Charging Station, CSMS, External System
	Scenario description	External setpoint control granted by CSMS. 1. CSMS has sent a SetChargingProfileRequest message with a *chargingProfilePurpose* `TxProfile` or `TxDefaultProfile` in which *chargingProfileKind* = `Dynamic` and with a *chargingSchedulePeriod* which has *operationMode* set to `ExternalSetpoint` (to allow controlling the setpoints) or `ExternalLimits` (to allow controlling charging/discharging limits). 2. During this *chargingSchedulePeriod* the External System is allowed to control the desired active power the EV should import or export. 3. If *chargingSchedule.duration* is set and the *setpoint/limit/dischargingLimit* is not updated by External System after *duration* seconds, then the *chargingSchedule* ends and Charging Station will fall back to the next valid charging profile. 4. If *chargingSchedule.duration* has not been set, then the *chargingSchedule* is valid indefinitely, until the charging profile is cleared or replaced by CSMS.
5	**Prerequisites**	Profile purpose must be `TxProfile` or `TxDefaultProfile` for V2X operations. Configuration variable `ExternalControlSignalsEnabled` = true. Configuration variable `ExternalConstraintsProfileDisallowed` = true.
6	**Post conditions**	If *chargingSchedule.duration* is set, then the use case ends when the setpoint is not updated after *duration* seconds, i.e. the schedule ceases to be valid when *duration* + *lastDynamicUpdate* > current time. If *chargingSchedule.duration* is not set, then the *chargingSchedule* remains valid until the charging profile is cleared or replaced by CSMS.
7	**Error Handling**	The value of *chargingSchedule.duration* acts as a timeout for updates from the external system. If no updates are received in time, the charging profile will fall back to the next valid charging profile.
8	**Remarks**	The communication method by which the External System communicates desired power levels to the Charging Station is out of scope of this document. Together with configuration variable `ExternalConstraintsProfileDisallowed` = false this use case controls when an external system is allowed to dictate setpoint and limits. The power setpoint communicated towards EV may never exceed *limit* or go below *dischargingLimit* of the composite (merged) charging schedules. (See V2X.03/04)

Q05 - External V2X setpoint control with a charging profile from CSMS - Requirements

Table 155. Q05 - Requirements

ID	Precondition	Requirements	Note
Q05.FR.01	When a ChargingSchedulePeriodType with *operationMode* = `ExternalSetpoint` in a `TxProfile` or `TxDefaultProfile` charging profile becomes active	Charging Station SHALL use the value from the External System for *setpoint* and *setpointReactive*.	During this period the values of *setpoint* and *setpointReactive* are given by External System. When applicable this also includes the L2 and L3 values of *setpoint(Reactive)*.
Q05.FR.02	When a ChargingSchedulePeriodType with *operationMode* = `ExternalLimits` in a `TxProfile` or `TxDefaultProfile` charging profile becomes active	Charging Station SHALL use the value from the External System for *limit* and *dischargingLimit*.	During this period the values of *limit* and *dischargingLimit* are given by External System. When applicable this also includes the L2 and L3 values of *(discharging)Limit*.
Q05.FR.03	Q05.FR.01 AND *chargingProfileKind* = `Dynamic`	External System MAY dynamically change *setpoint* and *setpointReactive* during the ChargingSchedulePeriodType.	When applicable this also includes the L2 and L3 values of *setpoint(Reactive)*.
Q05.FR.04	Q05.FR.02 AND *chargingProfileKind* = `Dynamic`	External System MAY dynamically change *limit* and *dischargingLimit* during the ChargingSchedulePeriodType.	When applicable this also includes the L2 and L3 values of *(discharging)Limit*.
Q05.FR.05	Q05.FR.03 AND External System updates *setpoint* or *setpointReactive*	Charging station SHALL update *setpoint* or *setpointReactive* in the charging profile and set *dynUpdateTime* to the current time.	
Q05.FR.06	Q05.FR.04 AND External System updates *limit* or *dischargingLimit*	Charging station SHALL update *limit* or *dischargingLimit* in the charging profile and set *dynUpdateTime* to the current time.	
Q05.FR.07	(Q05.FR.03 OR Q03.FR.04) AND *chargingSchedule.duration* is set AND current time > (*chargingSchedule.duration* + *dynUpdateTime*)	Charging Station SHALL consider the charging profile invalid and switch to using the next valid charging profile.	This is a fallback when External System is no longer responding within time set by *duration*.

Q06 - External V2X control with a charging profile from an External System

New in OCPP 2.1

No.	Type	Description
1	**Name**	External V2X control with a charging profile from an External System
2	**ID**	Q06
3	**Objective(s)**	An External System controls the charge and discharge limits or setpoint of a charging station via a `ChargingStationExternalConstraints` charging profile.
4	**Description**	An External System is not aware of individual charging sessions on a charging station. It can therefore only control charging station or EVSE power. This use case defines three scenarios: 1. Scheduled external limits control: a schedule of limits provided by external system. 2. Dynamic external limits control: limits are dynamically updated by external system. 3. Dynamic setpoint control: setpoint is dynamically updated by external system.
	Actors	EV, Charging Station, External System

No.	Type	Description
	Scenario #1	Scheduled external limits control. 1. External System sets a charging schedule with *limits* and/or *dischargingLimits*, which is represented in the charging station by a charging profile in which *chargingProfilePurpose* = `ChargingStationExternalConstraints`, *chargingProfileKind* = `Absolute` and with one or more *chargingSchedulePeriods* which have *operationMode* set to `ExternalLimits` and the externally received values for *limit* and *dischargingLimit*. 2. With these *chargingSchedulePeriods* the External System controls the limits of the active power to import or export. 3. When this external charging profile is set, charging station will send a NotifyChargingLimitRequest with *chargingLimitSource* = EMS and with the received schedule in *chargingSchedule* to CSMS. 4. If the charging profile is set to EVSE #0 (i.e. the whole charging station), then it is up to the charging station to divide charging and discharging power among the EVSEs to keep the total charging station power within *limit* and *dischargingLimit*.
	Scenario #2	Dynamic external limits control. 1. External System sets a single *(discharging)limit*, which is represented in the charging station by a charging profile in which *chargingProfilePurpose* = `ChargingStationExternalConstraints`, *chargingProfileKind* = `Dynamic` and with 1 *chargingSchedulePeriod* which has *operationMode* set to `ExternalLimits` and the externally received value for *limit* and/or *dischargingLimit*. 2. With this *chargingSchedulePeriod* the External System controls the limits of the active power to import or export. 3. Since *chargingProfileKind* = `Dynamic`, the EMS can change the *limit* and *dischargingLimit* dynamically without having to update the charging profile. 4. When this external charging profile is set and when the value of *limit/dischargingLimit* changes more than SmartChargingCtrlr.LimitChangeSignificance, the charging station will send a NotifyChargingLimitRequest with *chargingLimitSource* = EMS, *isDynamic* = true, and the charging schedule in *chargingSchedule* to CSMS. 5. If the charging profile is set to EVSE #0 (i.e. the whole charging station), then it is up to the charging station to divide charging and discharging power among the EVSEs to keep the total charging station power such that it remains within *limit* and *dischargingLimit*.
	Scenario #3	Dynamic external setpoint control. 1. External System sets a single *setpoint* and/or *setpointReactive*, which is represented in the charging station as a charging profile in which *chargingProfilePurpose* = `ChargingStationExternalConstraints`, *chargingProfileKind* = `Dynamic` and with 1 *chargingSchedulePeriod* which has *operationMode* set to `ExternalSetpoint` and a value for *setpoint* and/or *setpointReactive*. 2. With this *chargingSchedulePeriod* the External System controls the setpoint of the active and reactive power to import or export. 3. When this external charging profile is set and when the value of *limit/dischargingLimit* changes more than SmartChargingCtrlr.LimitChangeSignificance, charging station will send a NotifyChargingLimitRequest with *chargingLimitSource* = EMS, *isDynamic* = true, and with the received schedule in *chargingSchedule* to CSMS. 4. If the charging profile is set to EVSE #0 (i.e. the whole charging station), then it is up to the charging station to divide charging and discharging power among the EVSEs to ensure the total charging station power matches the *setpoint*.
5	**Prerequisites**	For discharging, at least one of the active charging sessions must have an active `TxProfile` or `TxDefaultProfile` for V2X operations. Configuration variable `ExternalControlSignalsEnabled` = true. Configuration variable `ExternalConstraintsProfileDisallowed` = false or absent.
6	**Post conditions**	Use case ends when the charging profile for `ChargingStationExternalConstraints` ends.
7	**Error Handling**	

No.	Type	Description
8	**Remarks**	The communication method by which the External System communicates the charging profile to the Charging Station is out of scope of this document. See section K. Smart Charging use cases K23 to K26 to see various topologies in which an external actor imposes charging limits. An external system (e.g. EMS) might not be aware (or does not have to be aware) of how many EVSEs the charging station has. In that case the `ChargingStationExternalConstraints` charging profile has to be assigned to EVSE #0. When a `ChargingStationExternalConstraints` charging profile is active with a *chargingSchedulePeriod* with *operationMode* = `ExternalSetpoint` and at the same time a `Tx(Default)Profile` charging profile is active with a *chargingSchedulePeriod* that also controls the setpoint (e.g. `CentralSetpoint` or `Central/LocalFrequency`) then the variable SmartChargingCtrlr.SetpointPriority determines which value of *setpoint* the charging station shall use. The power setpoint communicated towards EV may never exceed *limit* or go below *dischargingLimit* of the composite (merged) charging schedules. (See V2X.03/04) A scheduled external setpoint control, i.e. with *chargingProfileKind* = `Absolute` or `Relative`, is not mentioned as a scenario. It is possible, but less likely, since a setpoint is controlling the exact amount of power and an External System is unlikely to be able to predict a schedule for this.

Q06 - External V2X control with a charging profile from an External System - Requirements

Table 156. Q06 - Requirements

ID	Precondition	Requirements	Note
Q06.FR.01	When External System sends a schedule of limits	Charging Station SHALL represent this by a ChargingProfileType with *chargingProfilePurpose* = `ChargingStationExternalConstraints`, *chargingProfileKind* = `Absolute` and a *chargingSchedule* with *chargingSchedulePeriods* with *operationMode* = `ExternalLimits` representing the received schedule of *limit* and *dischargingLimit*.	When applicable this also include the L2 and L3 values of *(discharging)limit*.
Q06.FR.02	When External System sends a single limit value	Charging Station SHALL represent this by a ChargingProfileType with *chargingProfilePurpose* = `ChargingStationExternalConstraints`, *chargingProfileKind* = `Dynamic` and a *chargingSchedule* with one *chargingSchedulePeriod* with *operationMode* = `ExternalLimits` representing the received value of *limit* and *dischargingLimit*.	When applicable this also include the L2 and L3 values of *(discharging)limit*.
Q06.FR.03	When External System sends a schedule of setpoints	Charging Station SHALL represent this by a ChargingProfileType with *chargingProfilePurpose* = `ChargingStationExternalConstraints`, *chargingProfileKind* = `Absolute` and a *chargingSchedule* with *chargingSchedulePeriods* with *operationMode* = `ExternalSetpoint` representing the received schedule of *setpoint* and *setpointReactive*.	When applicable this also include the L2 and L3 values of *setpoint(Reactive)*.
Q06.FR.04	When External System sends a single setpoint value	Charging Station SHALL represent this by a ChargingProfileType with *chargingProfilePurpose* = `ChargingStationExternalConstraints`, *chargingProfileKind* = `Dynamic` and a *chargingSchedule* with one *chargingSchedulePeriod* with *operationMode* = `ExternalSetpoint` representing the received value of *setpoint* and *setpointReactive*.	When applicable this also include the L2 and L3 values of *setpoint(Reactive)*.

ID	**Precondition**	**Requirements**	**Note**
Q06.FR.05	Q06.FR.02	Charging Station SHALL use the value of *limit* and *dischargingLimit* that External System may vary dynamically within the ChargingSchedulePeriodType.	Also applies to L2 and L3 values.
Q06.FR.06	Q06.FR.04	Charging Station SHALL use the value of *setpoint* and *setpointReactive* that External System may vary dynamically within the ChargingSchedulePeriodType.	Also applies to L2 and L3 values.
Q06.FR.07	When for a charging profile with *chargingProfilePurpose* = `ChargingStationExternalConstraints` and *chargingProfileKind* = `Dynamic` the External System updates values within ChargingSchedulePeriodType	Charging Station SHALL update the value of *dynUpdateTime* to the current time.	
Q06.FR.08	When `ExternalConstraintsProfileDisallowed` = true AND an External System sends charging limits or setpoints	Charging Station SHALL refuse the settings and not create a charging profile for this.	This overrules Q06.FR.01 to Q06.FR.04.
NotifyChargingLimit			
Q06.FR.10	Q06.FR.01 OR Q06.FR.03	Charging Station SHALL send a NotifyChargingLimitRequest with *chargingLimitSource* = EMS and with the received schedule in *chargingSchedule* to CSMS.	
Q06.FR.11	Q06.FR.02 OR Q06.FR.04	Charging Station SHALL send a NotifyChargingLimitRequest with *chargingLimitSource* = EMS, *isDynamic* = true and with the received schedule in *chargingSchedule* to CSMS.	This *chargingSchedule* will only have a single period.
Q06.FR.12	Q06.FR.07 AND The value of *limit*, *dischargingLimit*, *setpoint*, *setpointReactive* changes more than SmartChargingCtrlr.LimitChangeSignificance	Charging Station SHALL send a NotifyChargingLimitRequest with *chargingLimitSource* = EMS, *isDynamic* = true and a schedule with the new values in *chargingSchedule* to CSMS.	Also applies to L2 and L3 values.
SetpointPriority			
Q06.FR.20	When a *chargingSchedulePeriod* in a charging profile with *chargingProfilePurpose* = `ChargingStationExternalConstraints` has *operationMode* = `ExternalSetpoint` AND A charging profile of *chargingProfilePurpose* = `TxProfile` or `TxDefaultProfile` also defines a *setpoint* or *setpointReactive*	Configuration variable SmartChargingCtrlr.SetpointPriority defines which charging profile the charging station shall apply.	
Q06.FR.21	Q06.FR.20 AND SmartChargingCtrlr.SetpointPriority = "ExternalControl" OR SmartChargingCtrlr.SetpointPriority does not exist	Charging station shall use the *setpoint* and *setpointReactive* from the active `ChargingStationExternalConstraints` profile.	This is default behavior for *operationMode* `ExternalSetpoint`.
Q06.FR.22	Q06.FR.20 AND SmartChargingCtrlr.SetpointPriority = "CSMS"	Charging station shall use the *setpoint* and *setpointReactive* from the `TxProfile` or `TxDefaultProfile` profile that is active, thus overruling the external setpoint.	Only `TxProfile` and `TxDefaultProfile` can contain a *setpoint(Reactive)*.
evseId = 0			
Q06.FR.30	When *chargingSchedulePeriod* has *operationMode* = `ExternalLimits` AND *evseId* = 0	Charging station SHALL keep the total active power of the charging station within *limit* and *dischargingLimit*.	

ID	Precondition	Requirements	Note
Q06.FR.31	When *chargingSchedulePeriod* has *operationMode* = `ExternalSetpoint` AND *evseId* = 0	Charging station SHALL try to match the total active power of the charging station with the value of *setpoint* and reactive power with *setpointReactive*.	It may not be possible to achieve this, as this highly depends on the value of the setpoints and the active sessions on the EVSEs.

Q07 - Central V2X control for frequency support

New in OCPP 2.1

No.	Type	Description
1	**Name**	Central V2X control for frequency support
2	**ID**	Q07
3	**Objective(s)**	To allow an EV to be used for frequency support, with control at the CSMS.
4	**Description**	The EV's charging and discharging behaviour is controlled by the CSMS for frequency support. This use case is intended for when calibrated frequency readings are required, and it is not economically viable to do these measurements locally. The Central system will constantly update the power setpoint (probably of an aggregated group of EV's) depending on the centrally measured frequency.
	Actors	EV, Charging Station, CSMS
	Scenario description	1. The CSMS has sent a SetChargingProfileRequest message with a *chargingProfilePurpose* `TxProfile` or `TxDefaultProfile` and a *chargingProfileKind* = `Dynamic`. The *chargingProfile* contains 1 *chargingSchedule* with 1 *chargingSchedulePeriod* which has *operationMode* set to `CentralFrequency` and *setpoint* to the desired active power the EV should import or export. 2. If *chargingSchedule.duration* is set on a `Dynamic` charging profile, then this specifies how long the *chargingSchedule* remains valid after SetChargingProfileRequest or UpdateDynamicScheduleRequest. 3. The Charging Station will obey the values of the (one and only) *chargingSchedulePeriod*, e.g. it will set the charge or discharge rate of the EV to *setpoint*. 4. CSMS updates the *chargingSchedulePeriod* periodically by sending a UpdateDynamicScheduleRequest.
	Scenario description #2	When the Charging Station does not receive a UpdateDynamicScheduleRequest message with a new schedule before the configured *duration* expires, then Charging Station shall continue with the next valid charging profile. If the Charging Station supports local frequency control, then this can be a `LocalFrequency` charging profile at a lower stack level. So, if no update from CSMS is received on time, then the Charging Station will switch from `CentralFrequency` to use case Q08 for local frequency control until a new UpdateDynamicScheduleRequest is received for the `CentralFrequency` charging profile.
5	**Prerequisites**	
6	**Post conditions**	Charging Profile ends or no more updates via UpdateDynamicScheduleRequest are received.
7	**Error Handling**	
8	**Remarks**	This use case is a special case of Q04 - Central V2X control with dynamic CSMS setpoint. This use case should not be confused with frequency support mechanisms described in section R. DER Control. In Q07 - Central V2X control for frequency support and Q08 - Local V2X control for frequency support the frequency support is the primary goal of the charging profile, whereas in the case of section R. DER Control the frequency support provided by frequency droop parameters or a frequency-Watt curve exists to satisfy grid code requirements.

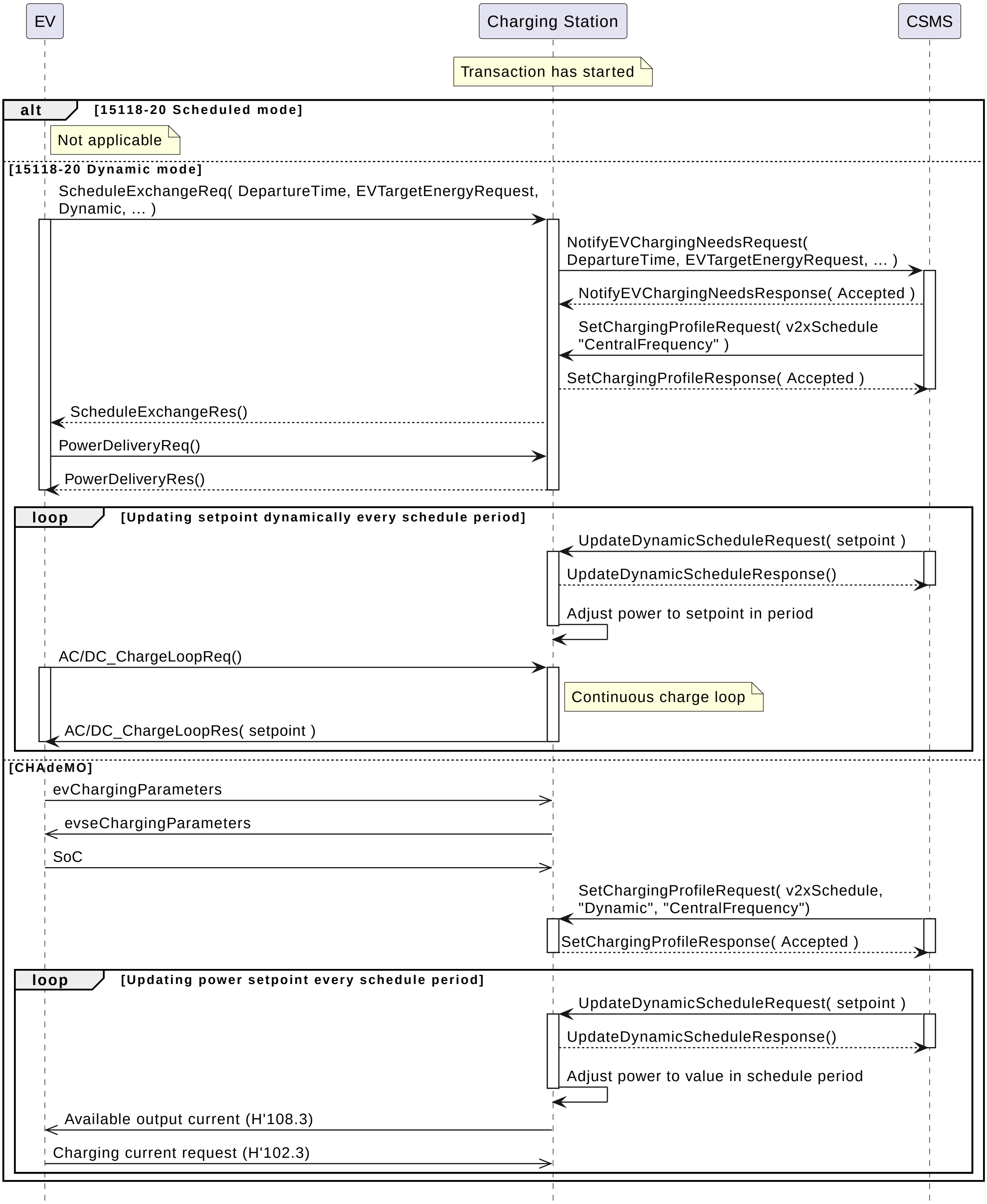

EV
Charging Station
CSMS
Transaction has started
alt [15118-20 Scheduled mode]
Not applicable
[15118-20 Dynamic mode]
ScheduleExchangeReq(DepartureTime, EVTargetEnergyRequest, Dynamic, ...)
NotifyEVChargingNeedsRequest(DepartureTime, EVTargetEnergyRequest, ...)
NotifyEVChargingNeedsResponse(Accepted)
SetChargingProfileRequest(v2xSchedule "CentralFrequency")
SetChargingProfileResponse(Accepted)
ScheduleExchangeRes()
PowerDeliveryReq()
PowerDeliveryRes()
loop [Updating setpoint dynamically every schedule period]
UpdateDynamicScheduleRequest(setpoint)
UpdateDynamicScheduleResponse()
Adjust power to setpoint in period
AC/DC_ChargeLoopReq()
Continuous charge loop
AC/DC_ChargeLoopRes(setpoint)
[CHAdeMO]
evChargingParameters
evseChargingParameters
SoC
SetChargingProfileRequest(v2xSchedule, "Dynamic", "CentralFrequency")
SetChargingProfileResponse(Accepted)
loop [Updating power setpoint every schedule period]
UpdateDynamicScheduleRequest(setpoint)
UpdateDynamicScheduleResponse()
Adjust power to value in schedule period
Available output current (H'108.3)
Charging current request (H'102.3)

Figure 192. V2X Operation Mode CentralFrequency

NOTE	This use case adheres to requirements related to CentralSetpoint from Q04 - Central V2X control with dynamic CSMS setpoint and Dynamic charging profiles from K01. There are no specific requirements for this use case.

Q08 - Local V2X control for frequency support

New in OCPP 2.1

No.	Type	Description
1	**Name**	Local V2X control for frequency support
2	**ID**	Q08
3	**Objective(s)**	To allow an EV to be used for frequency control, depending on local frequency readings.
4	**Description**	In this use case no setpoint is expected from the CSMS, and local frequency readings are always used to determine a setpoint by the Charging Station.
	Actors	EV, Charging Station
	Scenario description #1	Frequeny Containment Reserve (FCR) service 1. The CSMS has sent a SetChargingProfileRequest message, with a *chargingSchedule* that has one or more periods with *operationMode* set to `LocalFrequency` and a value for *v2xBaseline* and a *v2xFreqWattCurve* with the curve of power to be delivered or consumed depending on the frequency. 2. The Charging Station will continually get its latest frequency reading and calculate the new *setpoint* as *v2xBaseline* plus the interpolated power value from *v2xFreqWattCurve*.
	Scenario description #2	Automatic Frequency Restoration Reserve (aFRR) service 1. The CSMS has sent a SetChargingProfileRequest message, with a *chargingSchedule* that has one or more periods with *operationMode* set to `LocalFrequency` and a value for *v2xBaseline* and a *v2xFreqWattCurve* with the curve of power to be delivered or consumed depending on the frequency and a *v2xSignalWattCurve* with the power to be delivered or consumed depending on the signal from TSO. 2. The CSMS sends a AFRRSignalRequest message with a *signal* and *timestamp* as result of a signal received from TSO. The value of *signal* is applied when time is greater or equal to *timestamp*. 3. The Charging Station will continually get its latest frequency reading and calculate the new *setpoint* as *v2xBaseline* plus the interpolated power value from *v2xFreqWattCurve* and *v2xSignalWattCurve*.
5	**Prerequisites**	The field *limitAtSoC* is absent in *chargingSchedule*, or SoC of EV is still below the value in *limitAtSoC*.

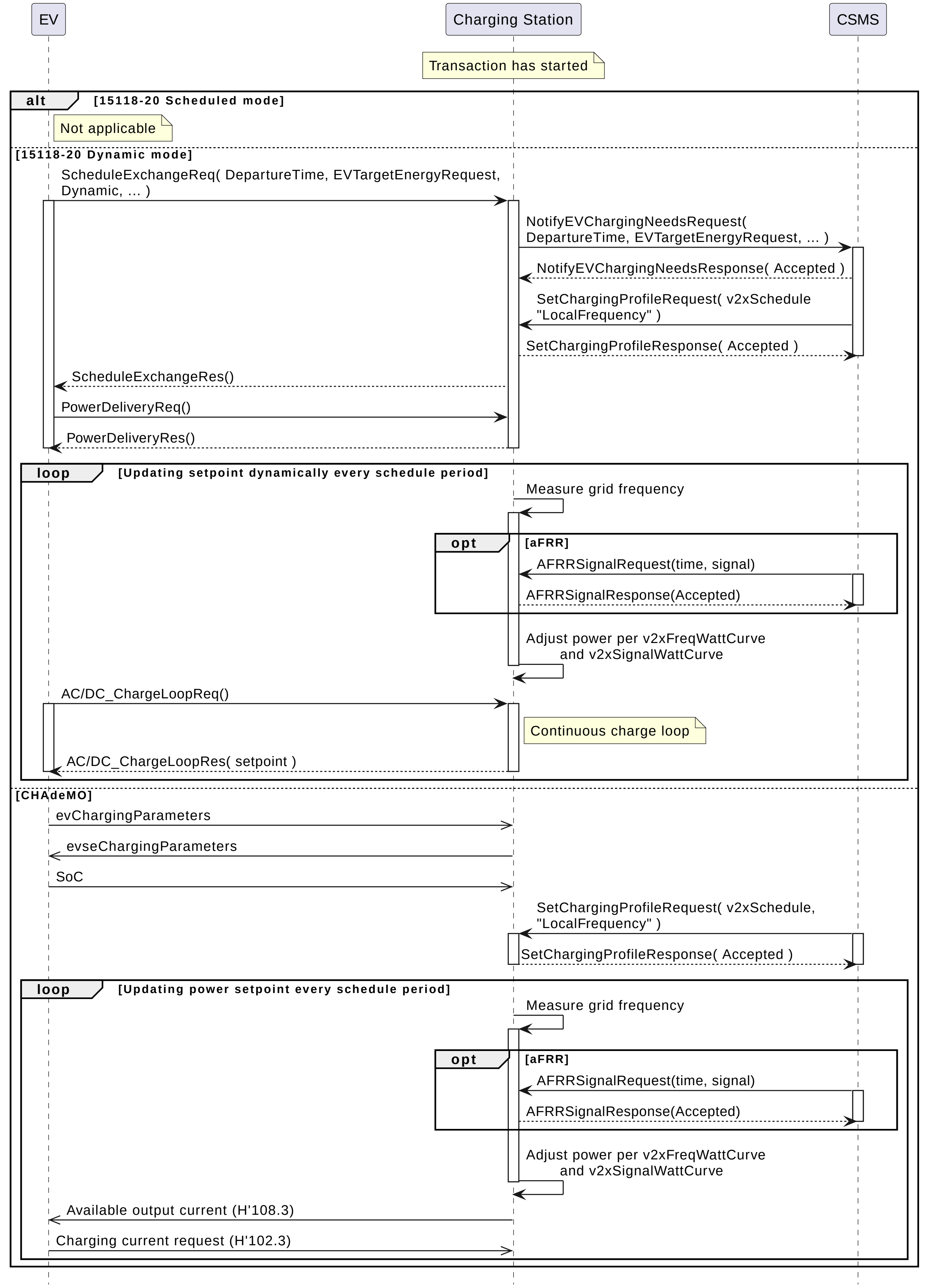

EV
Charging Station
CSMS
Transaction has started
alt
[15118-20 Scheduled mode]
Not applicable
[15118-20 Dynamic mode]
ScheduleExchangeReq(DepartureTime, EVTargetEnergyRequest, Dynamic, ...)
NotifyEVChargingNeedsRequest(DepartureTime, EVTargetEnergyRequest, ...)
NotifyEVChargingNeedsResponse(Accepted)
SetChargingProfileRequest(v2xSchedule "LocalFrequency")
SetChargingProfileResponse(Accepted)
ScheduleExchangeRes()
PowerDeliveryReq()
PowerDeliveryRes()
loop
[Updating setpoint dynamically every schedule period]
Measure grid frequency
opt
[aFRR]
AFRRSignalRequest(time, signal)
AFRRSignalResponse(Accepted)
Adjust power per v2xFreqWattCurve and v2xSignalWattCurve
AC/DC_ChargeLoopReq()
Continuous charge loop
AC/DC_ChargeLoopRes(setpoint)
[CHAdeMO]
evChargingParameters
evseChargingParameters
SoC
SetChargingProfileRequest(v2xSchedule, "LocalFrequency")
SetChargingProfileResponse(Accepted)
loop
[Updating power setpoint every schedule period]
Measure grid frequency
opt
[aFRR]
AFRRSignalRequest(time, signal)
AFRRSignalResponse(Accepted)
Adjust power per v2xFreqWattCurve and v2xSignalWattCurve
Available output current (H'108.3)
Charging current request (H'102.3)

Figure 193. V2X Operation Mode LocalFrequency

Q08 - Local V2X control for frequency support - Requirements

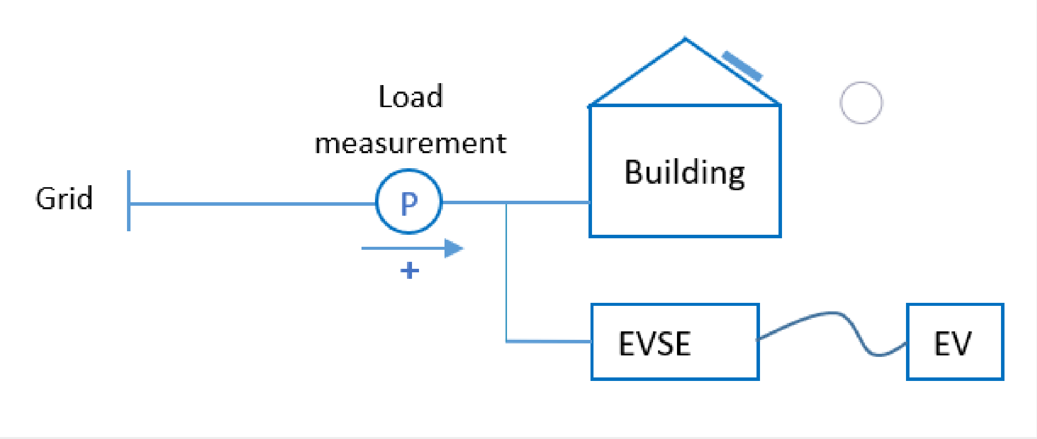
Table 157. Q08 - Requirements

ID.	Precondition	Requirements	Note
FCR			
Q08.FR.01	When a ChargingSchedulePeriodType has *operationMode* = `LocalFrequency`	The ChargingSchedulePeriodType SHALL NOT contain the fields *limit*, *dischargingLimit*, *setpoint*, *setpointReactive*.	This also applies to the L2 and L3 variants of these fields. The power setpoint is calculated from *v2xFreqWattCurve* and *v2xBaseline*.
Q08.FR.02	Q08.FR.01	The ChargingSchedulePeriodType SHALL have a *v2xFreqWattCurve* with at least two V2XFreqWattPointType elements, and a value for *v2xBaseline*.	
Q08.FR.03	When Charging Station receives a SetChargingProfile with a ChargingSchedulePeriodType that has *operationMode* = `LocalFrequency` AND The field *v2xBaseline* or *v2xFreqWattCurve* is missing or has less than two V2XFreqWattPointType elements	Charging Station SHALL respond with SetChargingProfileResponse with *status* = `Rejected` and *statusInfo.reasonCode* = "NoFreqWattCurve".	It is advised to mention the *chargingSchedule.id* and *chargingSchedulePeriod.startPeriod* in *statusInfo.additionalInfo* to tell which part of the charging schedule is invalid.
Q08.FR.04	When Charging Station receives a SetChargingProfile with a ChargingSchedulePeriodType that has *operationMode* = `LocalFrequency` AND The ChargingSchedulePeriodType contains any of the fields *limit*, *dischargingLimit*, *setpoint*, *setpointReactive*.	Charging Station SHALL respond with SetChargingProfileResponse with *status* = `Rejected` and *statusInfo.reasonCode* = "InvalidSchedule"	This also applies to the L2 and L3 variants of these fields. The power setpoint is calculated from *v2xFreqWattCurve* and *v2xBaseline*. It is advised to mention the *chargingSchedule.id* and *chargingSchedulePeriod.startPeriod* in *statusInfo.additionalInfo* to tell which part of the charging schedule is invalid.
Q08.FR.05	Q08.FR.01	The ChargingScheduleType that contains the ChargingSchedulePeriodType SHALL have *chargingRateUnit* = `W`.	A frequency-watt curve specifies power in Watts.
Q08.FR.06	When a ChargingSchedulePeriodType with *operationMode* = `LocalFrequency` starts	The Charging Station SHALL calculate the new *setpoint* as *v2xBaseline* plus the interpolated power value from *v2xFreqWattCurve* for the measured frequency.	
Q08.FR.07	During a ChargingSchedulePeriodType with *operationMode* = `LocalFrequency`, When the net frequency change in mHz is equal or more than V2XChargingCtrlr LocalFrequencyUpdateThreshold	The Charging Station SHALL calculate the new *setpoint* as *v2xBaseline* plus the interpolated power value from *v2xFreqWattCurve* for the measured frequency.	
aFRR			
Q08.FR.10	When a ChargingSchedulePeriodType has *operationMode* = `LocalFrequency` and contains field *v2xSignalWattCurve* AND Charging Station receives a AFRRSignalRequest with a *timestamp* and *signal*	Charging Station SHALL calculate a delta to the power setpoint based on the value of *signal* that it receives from AFRRSignalRequest and add that to the existing value of the power setpoint at time *timestamp*.	aFRR is added on top of the setpoint value calculated in Q08.FR.06/07

ID.	Precondition	Requirements	Note
Q08.FR.11	When Charging Station receives a AFRRSignalRequest and no ChargingSchedulePeriodType with *operationMode* = `LocalFrequency` and a field *v2xSignalWattCurve* is active	Charging Station SHALL respond with AFRRSignalResponse with *status* = `Rejected` and *statusInfo.reasonCode* = "NoSignalWattCurve"	It is advised to mention the *chargingSchedule.id* and *chargingSchedulePeriod.startPeriod* in *statusInfo.additionalInfo* to tell which part of the charging schedule is invalid.

Q09 - Local V2X control for load balancing

New in OCPP 2.1

No.	Type	Description
1	**Name**	Local V2X control for load balancing
2	**ID**	Q09
3	**Objective(s)**	To allow the EV to be utilized for locally controlled load balancing.
4	**Description**	This operation modes allows an EV to be utilized for load balancing controlled locally by the Charging Station. The Charging Station is able to read the grid meter of a building that consumes energy and possibly generates energy from, for example, solar panels.  Load measurement Grid P + Building EVSE EV
	Actors	EV, Charging Station, CSMS
	Scenario description	1. The CSMS has sent a SetChargingProfileRequest message, with a *chargingSchedule* that has one or more periods with *operationMode* set to `LocalLoadBalancing`. 2. The Charging Station will continually get the latest load power reading to calculate a new *setpoint* that will offset the measured power generation or consumption, such that it stays between the configured UpperThreshold and LowerThreshold. Optionally, there may be an UpperOffset and LowerOffset to control how much the power can exceed these thresholds.
5	**Prerequisites**	Charging Station is able to read the upstream meter.
6	**Post conditions**	ChargingSchedulePeriod ends
7	**Error Handling**	
8	**Remarks**	See section LocalLoadBalancing in the introduction of this chapter for more detailed information on how local load-balancing works. This use case only works for one Charging Station, but can work for multiple EVSEs when Charging Station has the capability to combine setpoints from multiple EVSEs to compensate the measured load at the upstream meter. Local load-balancing by multiple Charging Stations would have to be coordinated by a CSMS or Local Controller using the Central Setpoint use case.

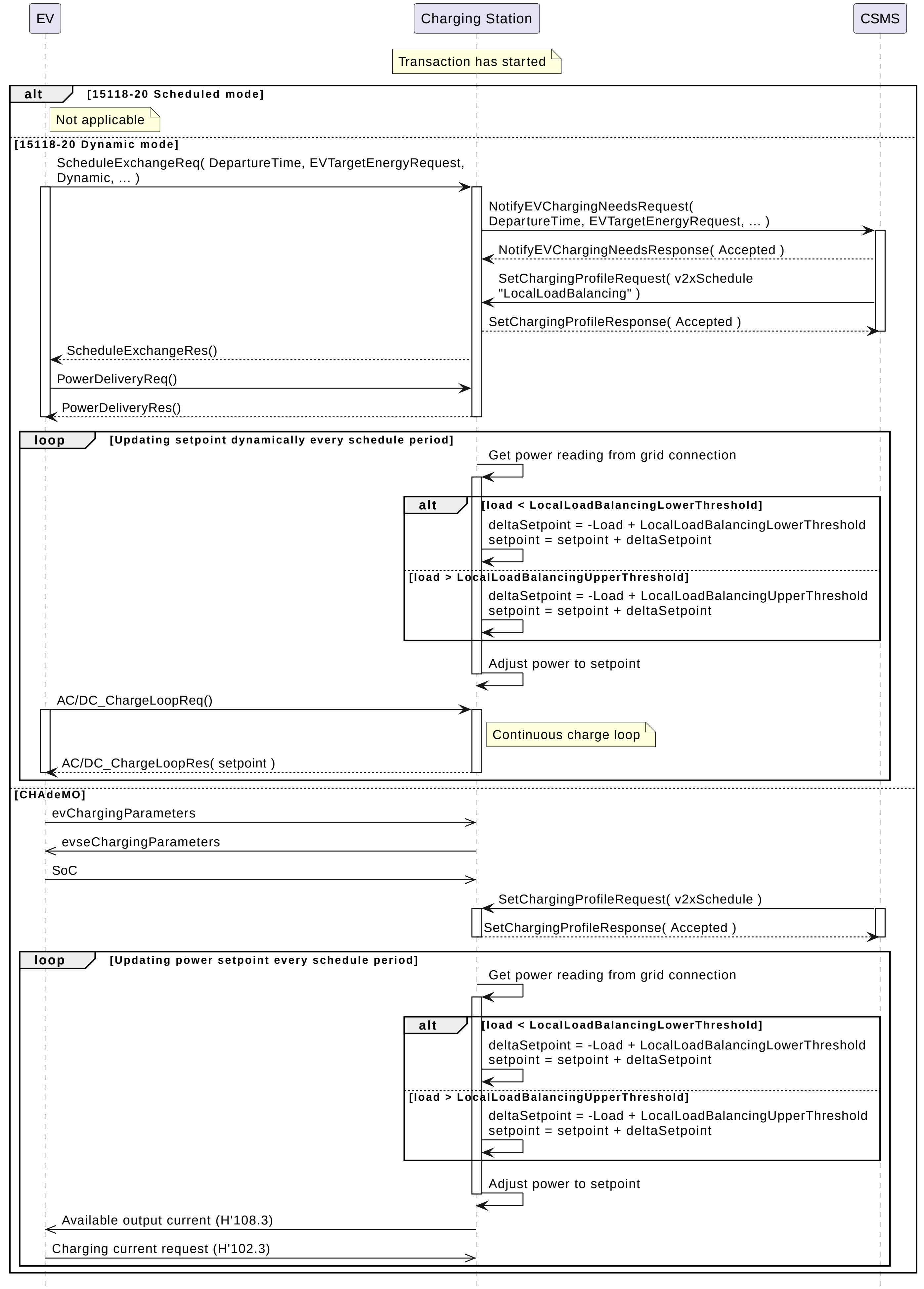

EV
Charging Station
CSMS
Transaction has started
alt [15118-20 Scheduled mode]
Not applicable
[15118-20 Dynamic mode]
ScheduleExchangeReq(DepartureTime, EVTargetEnergyRequest, Dynamic, ...)
NotifyEVChargingNeedsRequest(DepartureTime, EVTargetEnergyRequest, ...)
NotifyEVChargingNeedsResponse(Accepted)
SetChargingProfileRequest(v2xSchedule "LocalLoadBalancing")
SetChargingProfileResponse(Accepted)
ScheduleExchangeRes()
PowerDeliveryReq()
PowerDeliveryRes()
loop [Updating setpoint dynamically every schedule period]
Get power reading from grid connection
alt [load < LocalLoadBalancingLowerThreshold]
deltaSetpoint = -Load + LocalLoadBalancingLowerThreshold
setpoint = setpoint + deltaSetpoint
[load > LocalLoadBalancingUpperThreshold]
deltaSetpoint = -Load + LocalLoadBalancingUpperThreshold
setpoint = setpoint + deltaSetpoint
Adjust power to setpoint
AC/DC_ChargeLoopReq()
Continuous charge loop
AC/DC_ChargeLoopRes(setpoint)
[CHAdeMO]
evChargingParameters
evseChargingParameters
SoC
SetChargingProfileRequest(v2xSchedule)
SetChargingProfileResponse(Accepted)
loop [Updating power setpoint every schedule period]
Get power reading from grid connection
alt [load < LocalLoadBalancingLowerThreshold]
deltaSetpoint = -Load + LocalLoadBalancingLowerThreshold
setpoint = setpoint + deltaSetpoint
[load > LocalLoadBalancingUpperThreshold]
deltaSetpoint = -Load + LocalLoadBalancingUpperThreshold
setpoint = setpoint + deltaSetpoint
Adjust power to setpoint
Available output current (H'108.3)
Charging current request (H'102.3)

Figure 194. V2X Operation Mode LocalLoadBalancing

Q09 - Local V2X control for load balancing - Requirements

Table 158. Q09 - Requirements

ID.	Precondition	Requirements	Note
Q09.FR.01	When Charging Station receives a SetChargingProfileRequest with a ChargingSchedulePeriodType with *operationMode* = `LocalLoadBalancing` AND `V2XSupportedOperationModes` does not contain "LocalLoadBalancing"	Charging Station SHALL respond with SetChargingProfileResponse with *status* = `Rejected` and *statusInfo.reasonCode* = "UnsupportedParam".	
Q09.FR.02	When Charging Station receives a SetChargingProfileRequest with a ChargingSchedulePeriodType with *operationMode* = `LocalLoadBalancing` AND ((`V2XLocalLoadBalancing[UpperThreshold]` or `V2XLocalLoadBalancing[LowerThreshold]` are not set) OR (`V2XLocalLoadBalancing[UpperOffset]` or `V2XLocalLoadBalancing[LowerOffset]` are not set) OR (`V2XLocalLoadBalancing[UpperThreshold]` is not greater than `V2XLocalLoadBalancing[LowerThreshold]`))	Charging Station SHALL respond with SetChargingProfileResponse with *status* = `Rejected` and *statusInfo* = "MissingDevModelInfo"	
Q09.FR.03	During a ChargingSchedulePeriodType with *operationMode* = `LocalLoadBalancing`	Charging Station SHALL continously read the power reported by the upstream grid meter	
Q09.FR.04	Q09.FR.03	Charging Station SHALL calculate the power setpoint as follows:	"Load" is power from upstream meter. Positive values for consumption of energy.

```
If Load > UpperThreshold
    DeltaSetpoint =
        UpperThreshold
        + UpperOffset - Load
Else If Load < LowerThreshold
    DeltaSetpoint =
        LowerThreshold
        + LowerOffset - Load
Else
    DeltaSetpoint = 0
setpoint = setpoint
        + DeltaSetpoint
```

ID.	Precondition	Requirements	Note
Q09.FR.05	Q09.FR.03 AND *limit* and/or *dischargingLimit* are set	Charging Station SHALL cap the maximum (positive) value of *setpoint* to the value of *limit* and cap the minimum (negative) value of *setpoint* to the value of *dischargingLimit*.	This also applies to L2 and L3 values.

Q10 - Idle, minimizing energy consumption

New in OCPP 2.1

No.	Type	Description
1	**Name**	Idle, minimizing energy consumption
2	**ID**	Q10
3	**Objective(s)**	To request the EV to not perform any charging or discharging. Preconditioning of the vehicle is allowed.
4	**Description**	The CSMS requests EV to suspend any charging or discharging during one or more schedule intervals. CSMS requests EV to precondition itself in order to maintain the battery at an optimal temperature for charging or discharging.
	Actors	EV, Charging Station, CSMS
	Scenario description #1	When preconditioning of the battery is needed: 1. The CSMS sends a SetChargingProfileRequest message, with a *chargingSchedule* that has one or more periods with *operationMode* set to `Idle` and *preconditioning* set to True for this interval. 2. The Charging Station will request the EV to refrain from charging or discharging the battery, and to turn on preconditioning of the battery, which may, of course, draw some power from the EVSE.
	Scenario description #2	When preconditioning of the battery is not needed: 1. The CSMS sends a SetChargingProfileRequest message, with a *chargingSchedule* with *operationMode* set to `Idle` and *preconditioning* set to False for this interval. 2. The Charging Station will request the EV to refrain from charging or discharging and switch off preconditioning.
	Scenario description #3	Requesting EVSE to sleep: 1. The CSMS sends a SetChargingProfileRequest message, with a *chargingSchedule* that has one or more periods with *operationMode* set to `Idle` and *evseSleep* set to True for this interval. 2. The Charging Station will turn off the power electronics/modules associated with the current transaction. 3. When the EVSE is sleeping, any other charging command will “wake it up” (including limits/setpoints at 0 W). 4. If a TransactionEventRequest is sent while the EVSE is sleeping, then the field *evseSleep* in TransactionEventRequest will be true.
5	**Prerequisites**	
6	**Post conditions**	ChargingSchedulePeriod ends
7	**Error Handling**	When Charging Station does not support *evseSleep* = true, it will ignore it.
8	**Remarks**	This use case assumes a feature exists to request the EV to precondition the battery. It does not describe how CSMS or Charging Station do this. The configuration key SmartChargingCtrlr.SupportsEvseSleep has the value true if the option to request the EVSE to sleep is supported.

Q10 - Idle, minimizing energy consumption - Requirements

Table 159. Q10 - Requirements

ID.	Precondition	Requirements	Note
Q10.FR.01	When a *chargingSchedulePeriod* has *operationMode* = `Idle`	The fields *limit*, *dischargingLimit*, *setpoint* and *setpointReactive* SHALL be absent	This also applies to the L2/L3 variants of these fields.
Q10.FR.02	When Charging Station receives a SetChargingProfileRequest with a *chargingSchedulePeriod* that has *operationMode* = `Idle` AND any of the fields *limit*, *dischargingLimit*, *setpoint* and *setpointReactive* is present	Charging Station SHALL respond with SetChargingProfileResponse with *status* = `Rejected` and *statusInfo.reasonCode* = "InvalidSchedule".	It is advised to mention the *chargingSchedule.id* and *chargingSchedulePeriod.startPeriod* in *statusInfo.additionalInfo* to tell which part of the charging schedule is invalid.

ID.	Precondition	Requirements	Note
Q10.FR.03	When *chargingSchedulePeriod* has *operationMode* = `Idle` and *evseSleep* = true and `SupportsEvseSleep` is true	Charging Station SHALL turn off the electronics/modules associated with this transaction until an event happens that requires activation of these electronics/modules.	Events can be, for example, a change of *operationMode* in the next ChargingSchedulePeriod, receiving a new ChargingProfile (include one with limits of 0 W), stopping of the transction.
Q10.FR.04	Q10.FR.03	Charging Station SHALL report the value of *evseSleep* as true when sending TransactionEventRequests as long as the EVSE is sleeping.	
Q10.FR.05	When *chargingSchedulePeriod* has *operationMode* = `Idle` and *evseSleep* = true and `SupportsEvseSleep` is false or EVSE is not able to go into sleep mode	Charging Station SHALL report the value of *evseSleep* in TransactionEventRequests as false.	An absent field *evseSleep* is the same as having the value false.

Q11 - Going offline during V2X operation

New in OCPP 2.1

No.	Type	Description
1	**Name**	Going offline during V2X operation
2	**ID**	Q11
3	**Objective(s)**	To describe the amount of time that V2X operations may continue when the Charging Station is offline.
4	**Description**	If a Charging Station goes offline while a V2X operation is in progress, it is the field *maxOfflineDuration* in ChargingProfileType that determines how long the V2X operation may continue before the charging profile becomes invalid and Charging Station reverts to the next (lower) valid stack level.
	Actors	Charging Station, CSMS
	Scenario description	1. The Charging System is online and a V2X operation is active on the Charging System. 2. The Charging Station looses connection to the CSMS. 3. The Charging Station detects that it is offline. The *maxOfflineDuration* value counts from at this moment on. 4. The Charging Station reverts back to the next valid charging profile after *maxOfflineDuration* seconds. 5. The CSMS should assume that a V2X operation has reverted back to the next valid charging profile when it detects an offline situation. Note: since CS and CSMS may detect being offline at different moments, the CSMS should not try to guess the exact moment based on the *maxOfflineDuration* timeout value.
5	**Prerequisites**	The Charging Station must be able to detect an offline situation quickly. Typically, within 30 seconds.
6	**Post conditions**	V2X operation has reverted back to the next valid charging profile which has a longer or no *maxOfflineDuration*. (see K01.FR.101)
7	**Error Handling**	If there is no charging profile to fall back to after *maxOfflineDuration* seconds, charging continues without a charging profile, i.e. in "charging only" mode without limits.
8	**Remarks**	The next valid charging profile might also be a V2X profile if it has a value of *maxOfflineDuration* that has not yet elapsed. Charging profiles with *invalidAfterOfflineDuration* = true for which *maxOfflineDuration* has elapsed shall **not** be reactivated when the Charging Station is online again.

NOTE The requirements for this use case are covered by K01 requirements about *maxOfflineDuration*.

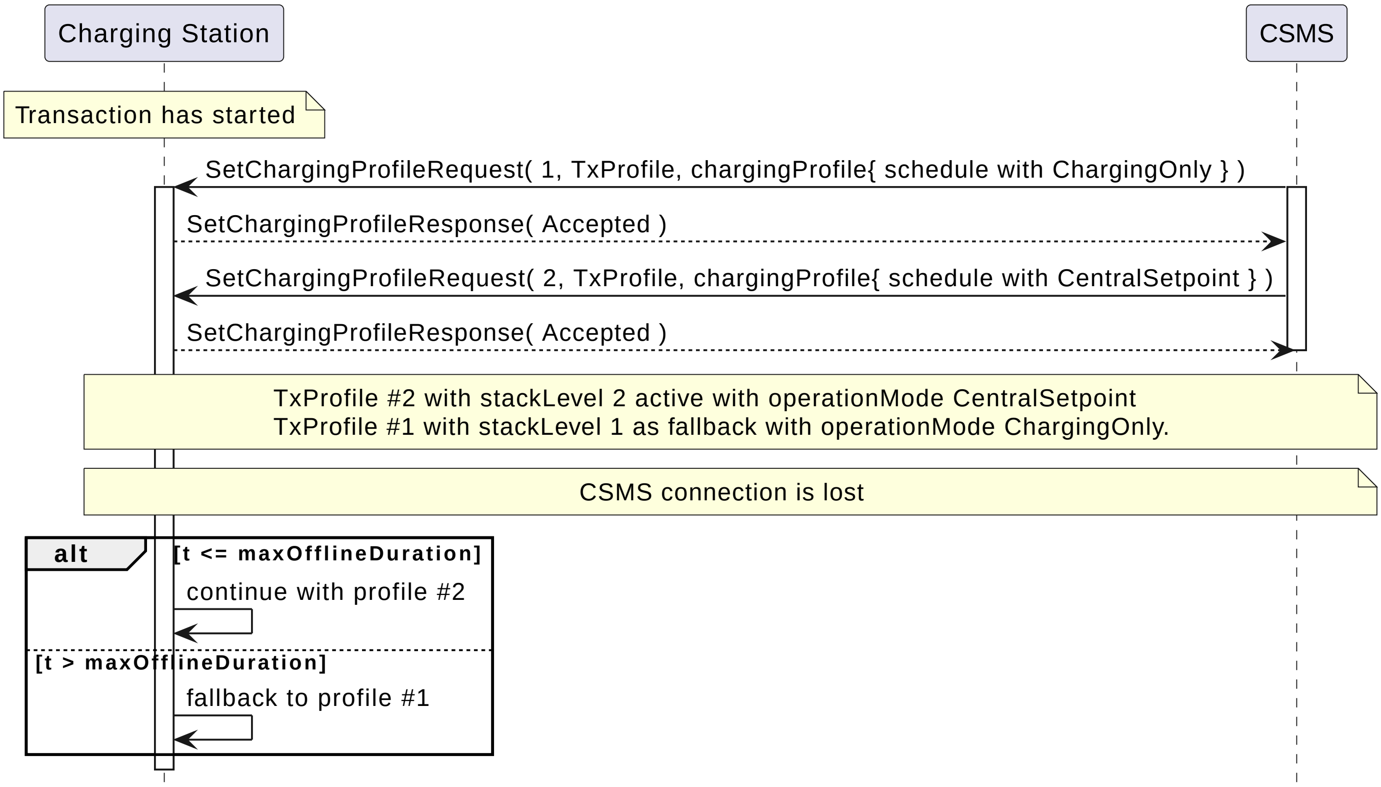

Charging Station
CSMS
Transaction has started
SetChargingProfileRequest(1, TxProfile, chargingProfile{ schedule with ChargingOnly })
SetChargingProfileResponse(Accepted)
SetChargingProfileRequest(2, TxProfile, chargingProfile{ schedule with CentralSetpoint })
SetChargingProfileResponse(Accepted)
TxProfile #2 with stackLevel 2 active with operationMode CentralSetpoint
TxProfile #1 with stackLevel 1 as fallback with operationMode ChargingOnly.
CSMS connection is lost
alt
[t <= maxOfflineDuration]
continue with profile #2
[t > maxOfflineDuration]
fallback to profile #1

Figure 195. V2X Operation Going Offline

Q12 - Resuming a V2X operation after an offline period

New in OCPP 2.1

No.	Type	Description
1	**Name**	Resuming a V2X operation after an offline period
2	**ID**	Q12
3	**Objective(s)**	To describe how to resume V2X operations after an offline period.
4	**Description**	The Charging Station is offline and a V2X charging operation is active on the Charging Station via a `TxProfile` or `TxDefaultProfile` charging profile. Charging Station has a regular (ChargingOnly) charging profile with stack level #1 and a V2X (CentralSetpoint) charging profile with stack level #2. If the offline period lasted less than *maxOfflineDuration* seconds, then the V2X operation mode will still be what it was before going offline, else it will have reverted back while offline to the next valid charging profile with a lower stack level (i.e. #1).
	Actors	Charging Station, CSMS
	Scenario description #1	*maxOfflineDuration* has not elapsed 1. The Charging Station restores connection to the CSMS. 2. The offline period lasted less than the value of *maxOfflineDuration* of the (V2X) charging profile. The charging profile therefore remained active during the offline period, and nothing changes in this respect when the connection is restored.
	Scenario description #2	*maxOfflineDuration* has elapsed, *invalidAfterOfflineDuration* = false 1. The Charging Station restores connection to the CSMS. 2. The offline period lasted longer than *maxOfflineDuration* of the (V2X) charging profile. This charging profile stopped being active after *maxOfflineDuration* seconds. 3. While offline, Charging Station selected the next valid charging profile with stack level #1 and is now using a charging profile that only has operation mode `ChargingOnly`. 4. When connection to CSMS is restored, Charging Station will re-evaluate its charging profiles and now select the charging profile with stack level #2 as the valid profile. 5. Charging Station will continue with V2X operations according to the charging profile.

No.	Type	Description
	Scenario description #3	*maxOfflineDuration* has elapsed, *invalidAfterOfflineDuration* = true 1. The Charging Station restores connection to the CSMS. 2. The offline period lasted longer than *maxOfflineDuration* of the (V2X) charging profile. This charging profile stopped being active after *maxOfflineDuration* seconds. 3. While offline, Charging Station selected the next valid charging profile with stack level #1 and is now using a charging profile that only has operation mode `ChargingOnly`. 4. When connection to CSMS is restored, Charging Station will re-evaluate its charging profiles. The charging profile with stack level #2 has *invalidAfterOfflineDuration* = true and is therefore no longer eligible. 5. Charging Station continues the charging profile with stack level #1 as the valid profile. 6. Charging Station will continue with charging-only operations according to the charging profile.
5	**Prerequisites**	Charging Station is offline
6	**Post conditions**	
7	**Remarks**	The requirements for this use case are covered by K01 - SetChargingProfile requirements about *maxOfflineDuration*.

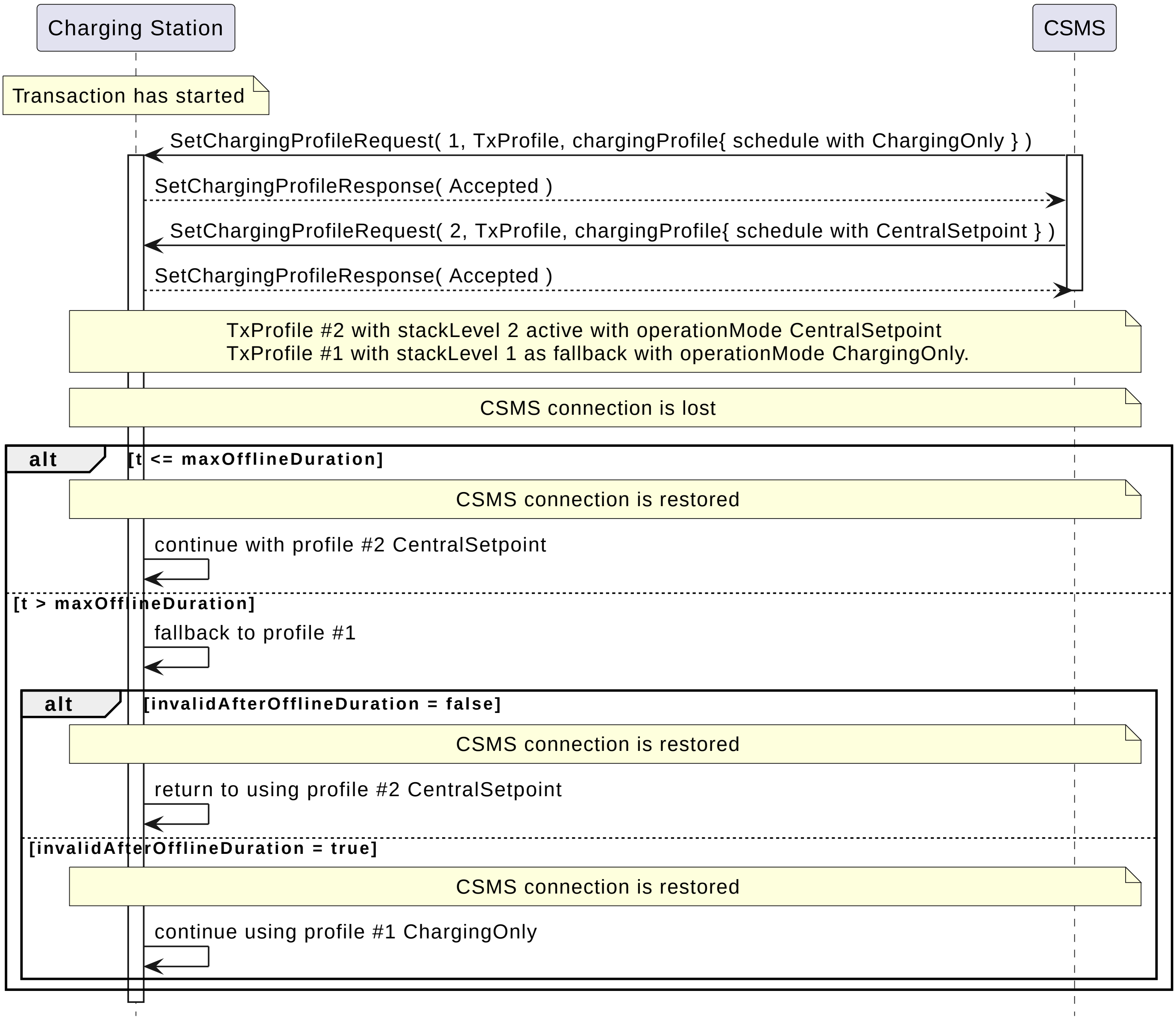

Charging Station
CSMS
Transaction has started
SetChargingProfileRequest(1, TxProfile, chargingProfile{ schedule with ChargingOnly })
SetChargingProfileResponse(Accepted)
SetChargingProfileRequest(2, TxProfile, chargingProfile{ schedule with CentralSetpoint })
SetChargingProfileResponse(Accepted)
TxProfile #2 with stackLevel 2 active with operationMode CentralSetpoint
TxProfile #1 with stackLevel 1 as fallback with operationMode ChargingOnly.
CSMS connection is lost
alt
[t <= maxOfflineDuration]
CSMS connection is restored
continue with profile #2 CentralSetpoint
[t > maxOfflineDuration]
fallback to profile #1
alt
[invalidAfterOfflineDuration = false]
CSMS connection is restored
return to using profile #2 CentralSetpoint
[invalidAfterOfflineDuration = true]
CSMS connection is restored
continue using profile #1 ChargingOnly

Figure 196. V2X Operation Resuming After Offline

R. DER Control

Chapter 1. Introduction

Utilities consider a charging station (or group of charging stations) that is performing bidirectional charging/discharging as a Distributed Energy Resource (DER). A DER needs to adhere to the local grid codes that govern how it should react in case of grid anomalies. This is controlled by a range of settings and curves that determine how to respond to frequencies or voltages that are too high or too low.

NOTE A utility may also require grid code support for charging-only Charging Stations, for example to provide frequency support. This chapter uses the term "DER control" for both bidirectional as well as charging-only Charging Stations.

Utilities usually require a DER to be registered and conform to certain regulations before it is permitted to provide energy to the grid. DC bidirectional (vehicle-to-grid) EVSEs could be registered in this way, because an EVSE is a stationary inverter for a battery pack (the EV). An AC bidirectional EV, however, uses its onboard inverter and is not stationary at all — it might charge and discharge at different locations. There is currently no regulation that covers the registration of AC bidirectional EVs.

In the US an EV capable of AC vehicle-to-grid shall be certified to the automotive standard SAE J3072 (which also calls out certification to IEEE 1547). However, as system operators want to have additional assurance that the EV follows the grid code, the EVSE shall also be certified to UL 1741, which allows the EVSE to monitor grid code compliance and disconnect the EV if the grid code is not followed. The SAE J3072 standard requires its own interface that connects to the inverter in the EV. There are at the time of publication (2024) few EV OEMs that have implemented this interface.

This specification assumes that the utility does not directly interface with charging stations or EVs for DER control, because the charging stations are only controlled by CSO, and the EVs have no interface with the utility. Instead, the CSO acts as an aggregator who controls (groups of) charging stations. The CSO, as aggregator, is assumed to take care of scheduling DER control messages towards charging stations when the utility sends (a schedule of) DER control messages.

The interface from CSO to a charging station is the OCPP protocol. OCPP 2.1 supports sending DER control messages, as described in this chapter.

Chapter 2. DER Control using OCPP and ISO 15118-20

2.1. CSO acting as aggregator

OCPP does not prescribe the protocol that must be used to communicate DER settings towards a CSO. The OCPP message set aims to provide support for DER control features from IEEE 1547 shown in Table 1. This table references two common protocols: IEC 61850, mostly used in Europe and Asia, and IEEE 2030.5, mostly used in the USA. In addition, IEEE 1547 D.3 mentions IEEE 1815 DNP3 and SunSpec Modbus as standardized interfaces that can be used for this. The fact that these are not referenced any further in this specification does not preclude them being used as a communication interface between utility and CSO for DER control.

Table 160. Comparing terminology across standards (based on Table C6 of [RefJ3072])

IEEE 1547-2018	**IEC 61850**	**IEEE 2030.5-2018**	**OCPP**
Mode/Function	**LN**	**DERControl**	**controlType**
Constant Power Factor	DFPF	opModFixedPFInject: Excit	FixedPFInject
Voltage - Reactive Power	DVVR	opModVoltVar: Curve	VoltVar
Active Power - Reactive Power	DWVR	opModWattVar: Curve	WattVar
Constant Reactive Power	DVAR	opModFixedVar: FixedVar	FixedVar
Voltage - Active Power	DVWC	opModVoltWatt: Curve	VoltWatt
High Voltage Trip Curve	DHVT	opModHVRTMustTrip: Curve	HVMustTrip
Low Voltage Trip Curve	DLVT	opModLVRTMustTrip: Curve	LVMustTrip
High Frequency Trip Curve	DHFT	opModHFRTMustTrip: Curve	HFMustTrip
Low Frequency Trip Curve	DLFT	opModLFRTMustTrip: Curve	LFMustTrip
Frequency-Droop (HF)	DHFW	opModFreqDroop	FreqDroop
Frequency-Droop (LF)	DLFW	opModFreqDroop	FreqDroop
Enter Service	DCTE	DefaultDERControl: setES…	EnterService
Cease to Energize and Trip	DCTE	opModEnergize	ChargingProfile
Limit Active Power	DWMX	opModMaxLimW: PerCent	LimitMaxDischarge
NA	DTCD	opModFixedW: SignedPerCent	ChargingProfile
NA	DWGC	opModFixedW: SignedPerCent	ChargingProfile

It is important to realize, that the OCPP messages do not provide one-to-one replacement of IEC 61850 or IEEE 2030.5 messages. It is assumed that the CSO performs an aggregator role towards the utility. The utility connects with the CSMS of the CSO, using an agreed protocol (e.g. one of IEC 61850, IEEE 2030.5, DNP3, Modbus). CSMS will then forward the appropriate information to the impacted charging stations. The burden of scheduling and prioritizing of messages lies mostly with CSMS.

For example, a utility can schedule a series of events with a start time and duration for a DER. A DER can be part of multiple groups, and one group can have a higher priority than another group. CSMS will receive these events and construct a timeline of events in which higher priority events supersede events with lower priority. CSMS will then send the appropriate OCPP messages to the affected charging station. ISO 15118-20 Amendment 1 can be used to transfer the DER control parameters to the EV, guaranteeing the EV will comply with the grid code requirements during times when the EV is feeding back to the grid to satisfy a request coming from the CSMS or an EMS. This is illustrated in the figure below.

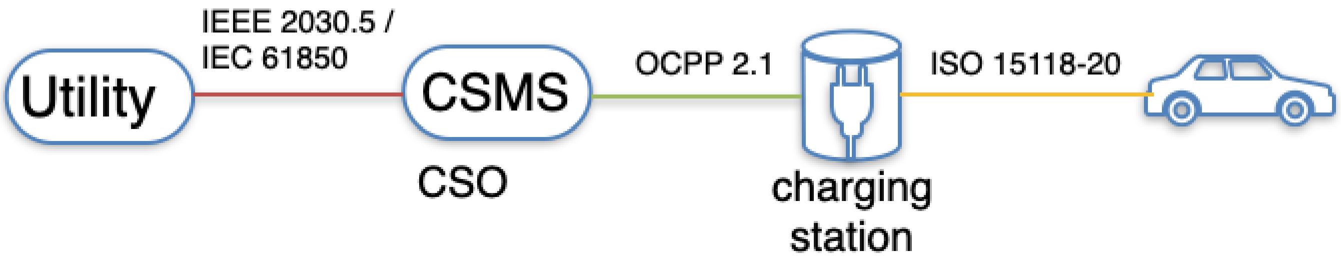


Figure 197. Protocol chain for DER control

2.2. DER Control for DC bidirectional power transfer

In case of a DC charging station, the AC-DC inverter is part of the EVSE in the charging station. The bidirectional EVSE is therefore the device that is acting as the DER that needs to be configured such that it complies to the grid codes.

In most cases the DER configuration will be static, since grid codes are not changed often, but it is possible for the CSO to dynamically update the DER settings of charging stations.

2.3. DER Control for AC bidirectional power transfer

In case of an AC charging station, the AC-DC inverter is part of the EV. As such it cannot be controlled directly via OCPP, because there is no OCPP connection with the EV. The charging station has its own communication protocol with the EV. This document assumes ISO 15118-20 as the communication protocol with the EV, but other protocols that support bidirectional charging, such as CHAdeMO, can also be used, although the number of parameters that can be exchanged may differ.

The current version of ISO 15118-20 for AC (ProtocolNamespace = "urn:iso:std:iso:15118:-20:AC", VersionNumberMajor = 1) available in 2023 does not provide support to configure the EV inverter with specific grid code settings, such as fault ride-through and frequency droop curves, or reactive power capabilities. However, the ChargeLoop message of ISO 15118-20, that is exchanged between charging station and EV several times a second, provides a lot of control over the charging. A charging station can use this to dynamically control the charging and discharging such that it satisfies the grid code requirements.

For example, a frequency droop curve can be implemented by the charging station. Based on the grid frequency this curve will tell how much discharging needs to be reduced or increased to help stabilize the grid frequency. The charging station adjusts the charging or discharging rate of the EV via the ChargeLoop messages. There is some latency involved, of course, and reaction speed highly depends on how frequent the EV sends its ChargeLoop messages. It will not suffice for settings that require a reaction time in the order of milliseconds.

Amendment 1 of ISO 15118-20 provides a new service "AC_BPT_DER" with capabilities to enforce grid code compliance. Two types of architectures are supported. An architecture in which the AC inverter in the EV is fully responsible for grid code compliance, and an architecture in which the responsibility is split between EV and Charging Station.

2.4. ISO 15118 V2G-AC inverter architecture

In the V2G-AC inverter architecture, the advanced smart inverter in the EV is fully responsible for grid code compliance. When the EV connects to the Charging Station, the latter will require a list of settings and supported DER controls that the EV must support when accepting the AC_BPT_DER service. Charging Station will then send all grid code settings to the EV. The grid code settings will have been sent to the Charging Station by the CSMS at some earlier point in time.

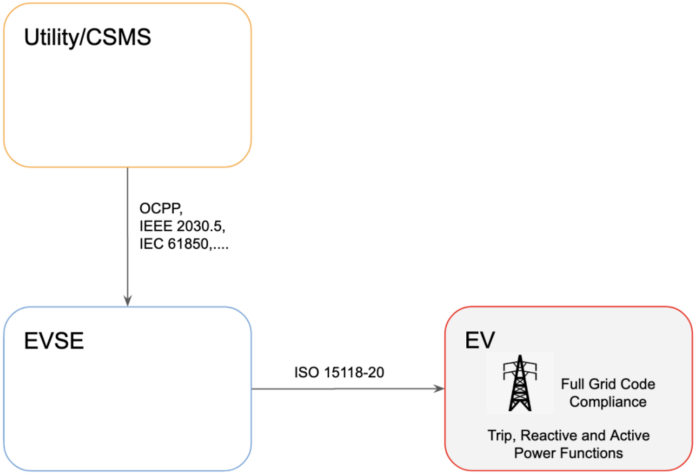

Utility/CSMS
OCPP,
IEEE 2030.5,
IEC 61850,....
EVSE
ISO 15118-20
EV
Full Grid Code Compliance
Trip, Reactive and Active Power Functions

Figure 198. V2G-AC inverter archicture (ISO 15118-20)

2.5. ISO 15118 V2G-Split inverter architecture

In the V2G-Split architecture the responsibility for grid code compliance is split between EV and Charging Station.

When the EV connects to the Charging Station, the latter will require a list of minimally required settings and DER controls that the EV must support. (This could even be an empty list.) Any DER controls that are required to comply to the grid code, but are not supported by the EV will be executed by the EVSE of the Charging Station. One of the possible configurations is the one represented in the diagram, where the EVSE ensures grid code compliance by measuring frequency and voltage deviations and calculating the appropriate active and/or reactive power target values. The EV will receive the targets requested by the EVSE and consume or inject active/reactive power. Both the EV and EVSE or just one shall also have interface protection means, which will be triggered according to the trip functions. The EV may perform reactive and active power functions independently, or receive target requests by the EVSE to consume or inject active/reactive power.

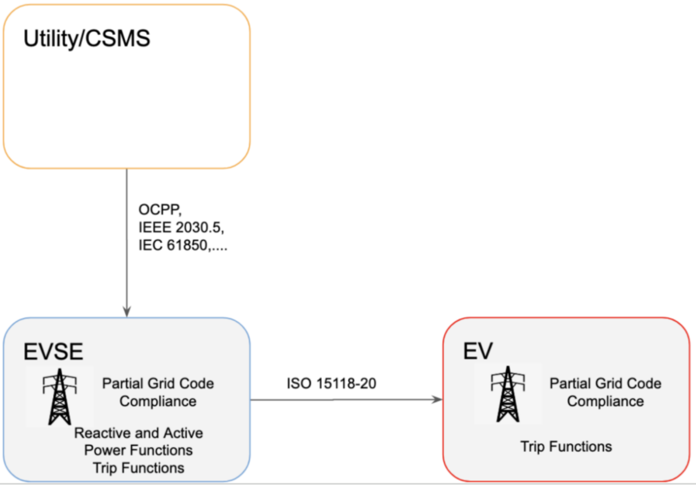

Utility/CSMS
OCPP,
IEEE 2030.5,
IEC 61850,....
EVSE
Partial Grid Code Compliance
Reactive and Active Power Functions
Trip Functions
ISO 15118-20
EV
Partial Grid Code Compliance
Trip Functions

Figure 199. V2G-Split inverter architecture (ISO 15118-20)

NOTE From an OCPP point of view the inverter architecture, V2G-AC or V2G-Split, is irrelevant. CSMS sends the DER controls to the Charging Station and does not need to know if the inverter is part of a DC Charging Station or AC Charging Station with V2G-AC or V2G-Split architecture.

2.6. DER nameplate information

The DER nameplate information is information about capabilities and make and model of the inverter. For a DC charging station this information can be provided by the CSO, who is aware of the station’s inverter capabilities. A CSO will either have this information based on the make and model of the charging station, or the CSO is able to request the data from the charging station by retrieving data about the AcDcConverter component of the EVSE in the charging station.

When the ISO 15118-20 communication is established using the AC_BPT_DER service, the vehicle will send its inverter capabilities, including the manufacturer, model and version.

The capabilities of the inverter of the EVSE do not have to match the capabilities of the vehicle. The combined capabilities of station and vehicle can only be given once the ISO 15118-20 ChargeParameterDiscovery process has completed. The results are sent to CSMS via the OCPP NotifyEVChargingNeedsRequest message.

2.7. DER controls

The behavior of a DER (i.e. charging station or EV inverter) can be controlled by so-called DER controls. These are settings with parameters to control active or reactive power, other settings, such as a specific power factor and ramp rates, and curves that control behavior.

NOTE ISO 15118-20 sends a complete set of DER controls to the EV each time it requests to use the AC_BPT_DER service for AC bidirectional charging. This is because the EV is seen as a "new" DER resource that needs to be fully configured. Standards like IEEE 2030.5 and IEC 61850, however, allow a utility to submit or update individual controls. A utility might, for example, for the set of voltage trip curves that are installed on a DER, only update the overvoltage trip curve. This approach makes sense for static DER resources, such as solar panels and DC charging stations. OCPP messages are designed for controlling individual DER controls, because that is what utilities or aggregators are likely to provide. A charging station will group the set of active DER controls into a single message with voltage trip curves, frequency trip curves, reactive power support, etc. for sending to the EV.

Power settings

If a utility wishes to state (reactive) power setpoints or limits, this can be achieved via the CSO. Setting of active or reactive power setpoints via CSMS is already supported by OCPP charging profiles. No new messages towards charging station are needed to support this. Instead, a power setpoint or limit will be translated to an OCPP charging profile towards the stations.

Parameter settings

Other control settings, such as a fixed power factor, frequency droop, are transferred to the charging station via a SetDERControlRequest message. The settings can be scheduled with a start time and duration. The currently active settings can be retrieved with a GetDERControlRequest message.

The following settings can be set via SetDERControlRequest:

1. Voltage trip curves
2. Frequency trip curves
3. Enter service parameters
4. Reactive power curves and parameters
5. Active power curves and frequency droop

Frequency droop is a parameterized form of the Freq-Watt curve that is determined by slope and a dead-band parameters.

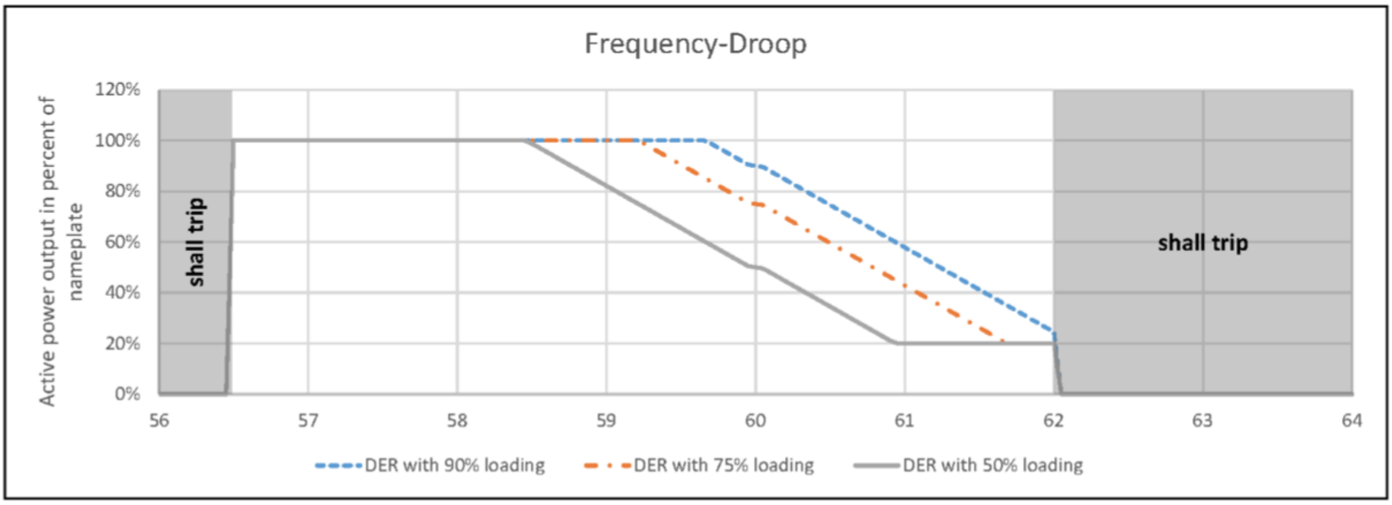


Figure 200. Example Frequency Droop curve from IEEE 1547-2018

Curves

DER curves describe behavior that a charging station must perform **autonomously** in case of a grid anomaly. Such curves are configured on the charging station via the OCPP SetDERControlRequest messages.

While only one curve per curve-type can be active at the same time, different curve-types can be active at the same time if they do not conflict. These curves are used to provide autonomous control in a predictable fashion. For example, assuming a volt-watt curve is active; if the inverter senses an over-voltage situation a volt-watt curve would direct the inverter to lower its power output during discharging. Likewise, in an under-voltage situation, the same curve would likely direct the DER to increase its output during discharging.

Trip and ride-through curves

A ride-through curve describes how long a DER must stay connected when frequency of voltage deviates from nominal. A trip curve describes after how many seconds a DER must disconnect when frequency of voltage deviates from nominal. Both curves are called "trip curves" in this document, where a "may trip" curve defines the ride-through behavior and a "must trip" curve defines the trip behavior.

Trip curves exist for high and low frequency or voltage anomalies:

1. High Frequency Must/May Trip
2. Low Frequency Must/May Trip
3. High Voltage Must/May Trip and Momentary Cessation
4. Low Voltage Must/May Trip and Momentary Cessation

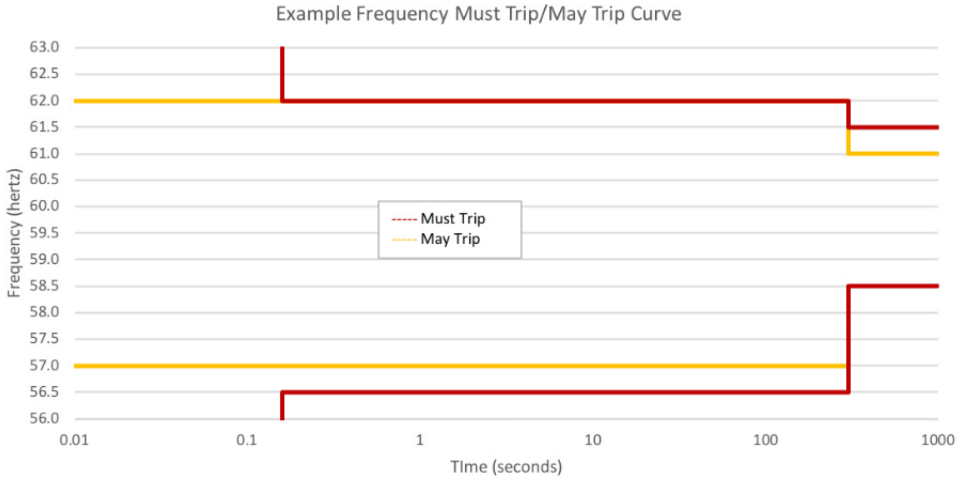


Figure 201. Example frequency trip curves from IEEE 2030.5

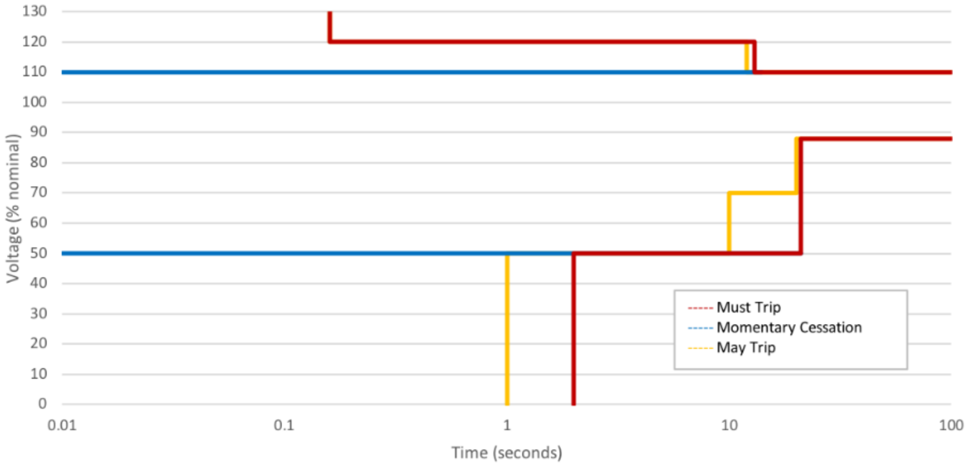


Figure 202. Example voltage trip curves from IEEE 2030.5

Volt and Watt curves

The following is a list of (reactive) power curves based on the measured frequency, voltage or power:

1. Frequency-Power curve, P(F)
2. Voltage-Reactive Power curve, Q(U)
3. Voltage-Power curve, P(U)
4. Power-Power Factor curve, Pf(P)
5. Power-Reactive Power curve, Q(P)

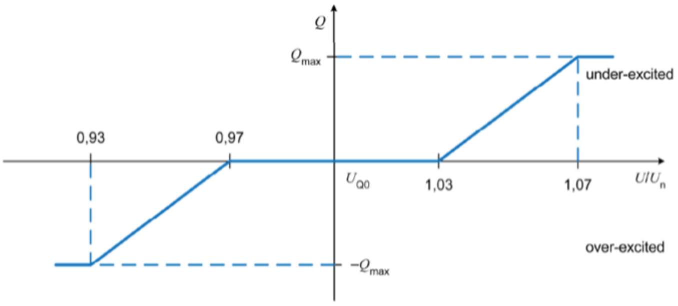


Figure 203. Example Q(U) curve used in Germany (VDE-AR-N 4105)

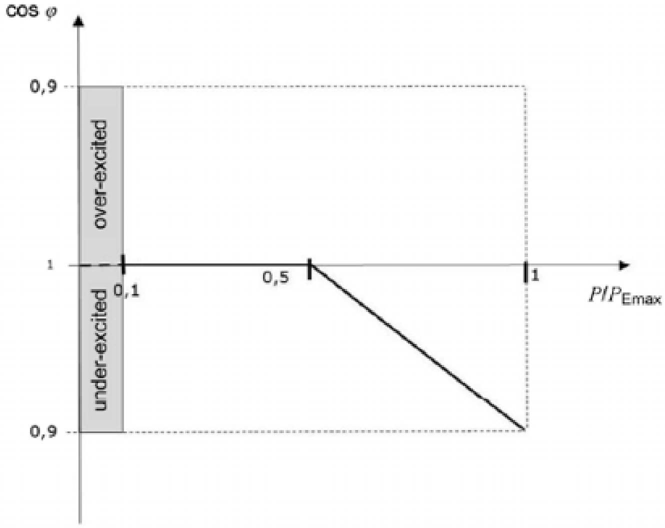


Figure 204. Example power factor curve, cos phi(P), used in Germany (VDE-AR-N 4105)

2.8. DER alarms

When a charging station starts to deviate from normal behavior, because it is forced to follow a DER curve or setting, then this will be signalled to CSMS, such that the CSO is aware of this situation. Similarly, the charging station will send a signal when this event has ended.

This is supported via the OCPP NotifyDERAlarmRequest message, which tells which DER control is taking over and what caused it. The message is sent again with an "ended" flag to signal when the event has ended.

2.9. Prioritizing of controls

There are two mechanisms that affect the priority of DER controls. Controls can be DER curves or parameter settings in this context. Firstly, there are default DER controls and scheduled ones. Secondly, there are different priorities as defined by the *priority* field of the setting. A *priority* = 0 is the highest priority.

Default controls

Default DER controls are active when there is no scheduled DER control of the same type that is active at that point in time. "Type" in this context means a DER control parameter, such as `FreqDroop` or a DER curve, such as `FreqWatt` or `HFMustTrip`. There may be scheduled DER controls installed, but they will not become active until their specified *startTime*, and stop being active after *duration* seconds after *startTime*, after which the system falls back to the default DER control (if any).

Priority

Only one control of a specific type a can be active at any one time. If more than one control of the same type is installed, then the one with the highest priority (i.e. the lowest *priority* value!) will prevail. This applies to default and scheduled DER controls: the default control with the lowest *priority* value will be used as default control, and the scheduled control with the lowest *priority* value will be applied at its start time.

The prioritization of a scheduled DER controls takes place at the moment when the message is received. When a DER control is received with a lower *priority* value than other controls of the same type, then these other controls become superseded **immediately**, as long as they have not yet started. This happens even when the new control has a much later start time than the existing ones. If a to-be-superseded control has already started (i.e. it is past *startTime*), then it will remain active and only become superseded at the *startTime* of the superseding control. A superseded control will no longer be used, but will remain to exist in the charging station until its duration expires.

This method of scheduling has deliberately been made to match the process for prioritization that is described in [RefCSIP]. The following examples from [RefCSIP] illustrate the behavior by describing two very similar overlapping event scenarios that only differ in when the charging station receives the controls. In the first case, the charging station receives both DER controls prior to the start of either. In this case, the charging station does **not execute** the lower priority (superseded) control at all. It only executes the higher priority control as shown in the figure below.

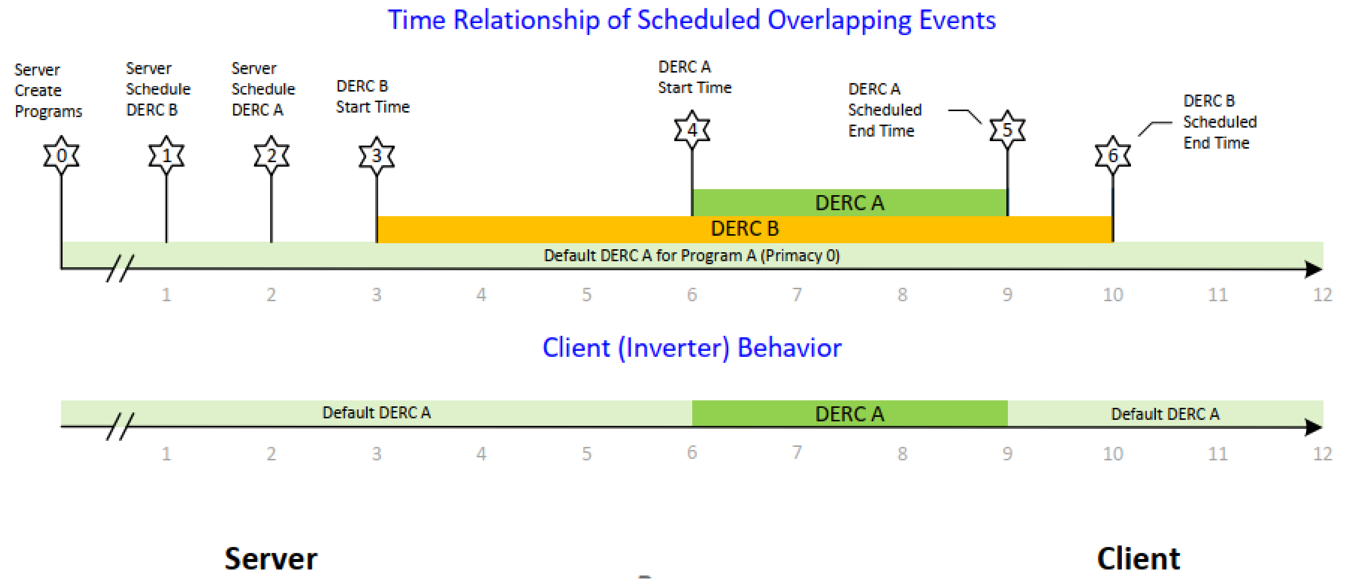


Figure 205. Control received before start of superseded control. From [RefCSIP]

In the second case, the charging station receives the higher priority control while executing the lower priority control. In this case, the charging station **continues** with the lower priority control until the start time of the higher priority control. It then supersedes the lower priority control and switches to executing the higher priority control as shown in the figure below.

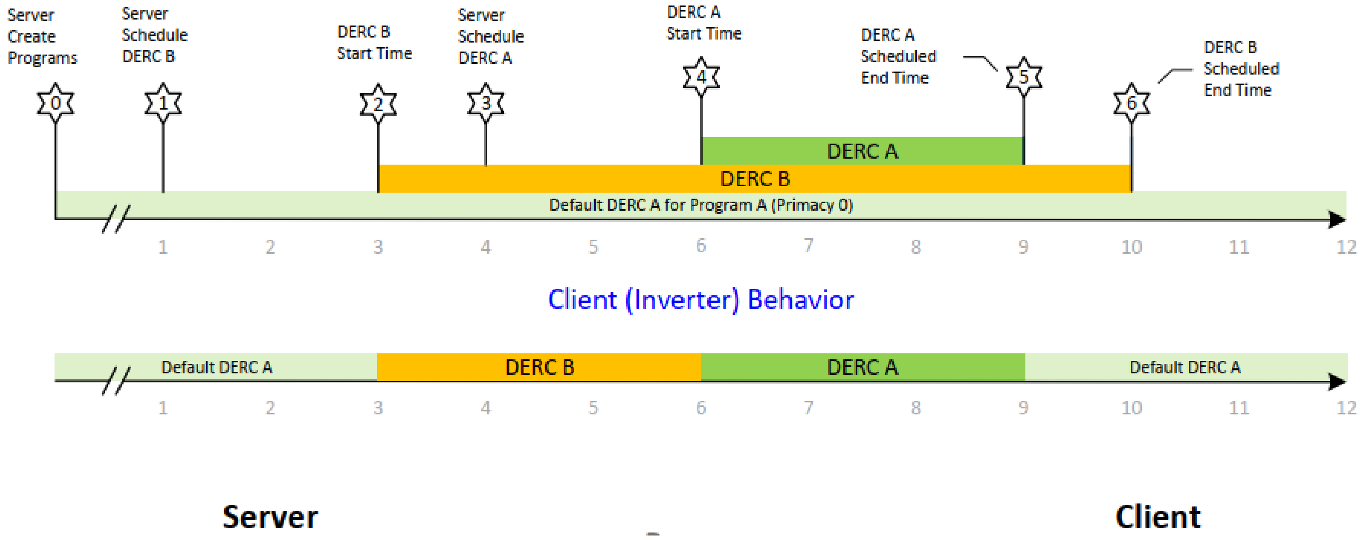


Figure 206. Control received after start of superseded event. From [RefCSIP]

Superseded controls remain present in the charging station and are reported with a *isSuperseded* flag set to true. Controls that are past their scheduled time and duration will be automatically removed by the charging station.

Scheduling by CSMS

Ideally, CSMS will take care of the prioritization and only send the highest priority DER control to the charging station. However, in some cases this must be handled by the charging station. An example of this is the situation where CSMS has already sent a DER control for "X" with *priority* = 1 to the charging station, and this control is now active. Then suddenly CSMS receives another DER control for "X" with *priority* = 0 that is overlapping, as shown in figure Control received after start of superseded event. From [RefCSIP]. If charging station would not be able to handle controls with different priority, then CSMS would have to send the new control with *priority* = 0 at the moment that it should start, thus replacing the older control. However, due to network and system latency, it would be impossible to guarantee that the new event starts exactly at its *startTime*. The charging station will thus have to deal with two or more controls for the same type with different priorities.

Chapter 3. Use Cases for DER control

Following use cases describe how a power setting, a DER setting, a trip curve and a power curve can be configured and activated.

R01 - Starting a V2X session with DER control in EVSE

New in OCPP 2.1

No.	Type	Description
1	**Name**	Starting a V2X session with DER control in EVSE
2	**ID**	R01
3	**Objective(s)**	To start a bidirectional charging session with DER control performed by inverter in the EVSE. This is always the case for DC EVSEs, but can also be used in case of AC charging if the Charging Station is capable of controlling (dis)charging behavior of EV such that it meets DER control settings.
4	**Description**	This use case illustrates how DER control can be executed in three different scenarios: ISO 15118-20 DC_BPT, CHAdeMO and ISO 15118-20 AC_BPT. A utility has set specific grid parameters (DER controls) that are required to be applied to bidirectional charging sessions. CSMS configures the settings at the Charging Station. An EV starts a bidirectional charging session. No DER control settings need to be applied to the EV, since the EVSE of the Charging Station will handle these. For an ISO 15118-20 DC_BPT or CHAdeMO charging session, the inverter is part of (and controlled by) the Charging Station. For an ISO 15118-20 AC_BPT charging session, the inverter is in the EV and cannot be configured by the Charging Station. Instead, the Charging Station will control charging behavior via ChargeLoop message in a way that it adheres as much as possible to the DER control settings of the Charging Station.
	Actors	Utility, CSMS, Charging Station, EV
	Scenario description #1	*A DC bidirectional session using ISO 15118-20 DC_BPT* 1. EV and Charging Station start an ISO 15118-20 session. 2. Upon successful authorization the EV starts service negotiation and selects the **DC_BPT** service. 3. EV and Charging Station exchange charge parameters 4. Charging Station sends NotifyEVChargingNeedsRequest with *departureTime* and *v2xChargingParameters*. 5. CSMS calculates a TxProfile for this session, which may contain periods of discharging, and sends it as a SetChargingProfileRequest. 6. Charging Station responds with ScheduleExchangeRes to EV. a. If *controlMode* = `Scheduled` is used, then ScheduleExchangeRes will contain the TxProfile schedule. b. If *controlMode* = `Dynamic` is used, then Charging Station will update charging/discharging levels via the ChargeLoop messages with EV to match the TxProfile schedule. 7. EV requests start of charging with a PowerDeliveryReq. 8. EV starts charging. 9. EV will start discharging according to schedule periods in TxProfile schedule. 10. DER control is executed by Charging Station. a. Discharging power of EVSE inverter will be limited to 50% of its rated capacity. b. Power will be adjusted according to grid voltage as defined by the configured VoltWatt curve.

No.	Type	Description
	Scenario description #2	*A DC bidirectional session using CHAdeMO* 1. EV is authorized and starts a charging session. 2. EV and Charging Station exchange charge parameters. 3. CSMS calculates a TxProfile for this session, which may contain periods of discharging, and sends it as a SetChargingProfileRequest. 4. Charging Station will update charging/discharging levels via CHAdeMO messages with EV to match the TxProfile schedule. 5. EV will start discharging according to schedule periods in TxProfile schedule. 6. DER control is executed by Charging Station: a. It will limit discharging power of EVSE inverter to 50% of its rated capacity. b. It will adjust power of EVSE inverter according to grid voltage as defined by the configured VoltWatt curve.
	Alternative description	*An AC bidirectional session using ISO 15118-20 AC_BPT* 1. EV and Charging Station start an ISO 15118-20 session. 2. Upon successful authorization the EV starts service negotiation and selects the **AC_BPT** service. *The following steps 3 to 9 are identical to the DC_BPT scenario* 10. Inverter is in EV, but DER control is executed by Charging Station: a. It will limit discharging power of EV to 50% of its rated capacity during ChargeParameterDiscovery by setting the EVSEMaximumDischargePower to 50% of reported EVMaximumDischargePower. b. It will adjust power according to grid voltage as defined by the configured VoltWatt curve by controlling EVSETargetActivePower using ChargeLoop control.
5	**Prerequisites**	• In advance utility has sent DER control settings to the CSMS of CSO. In this example these are a VoltWatt curve and a limitation of discharging power to 50% of the rated power of the inverter. • CSMS has conveyed these settings as SetDERControlRequest messages to the Charging Station: 1. SetDERControlRequest(isDefault = true, controlType = `VoltWatt`, curve = {…}) 2. SetDERControlRequest(isDefault = true, controlType = `LimitDischargePower`, limitMaxDischarge = { pctMaxDischargePower = 50.0}) See use cases R04 - Configure DER control settings at Charging Station.
6	**Post conditions**	The charging session adheres to DER control settings from utility.
7	**Error Handling**	
8	**Remarks**	The descriptions using ISO 15118 messages, such the use of ChargeParameterDiscovery and ChargeLoop for DER control, are only informative and not prescriptive.

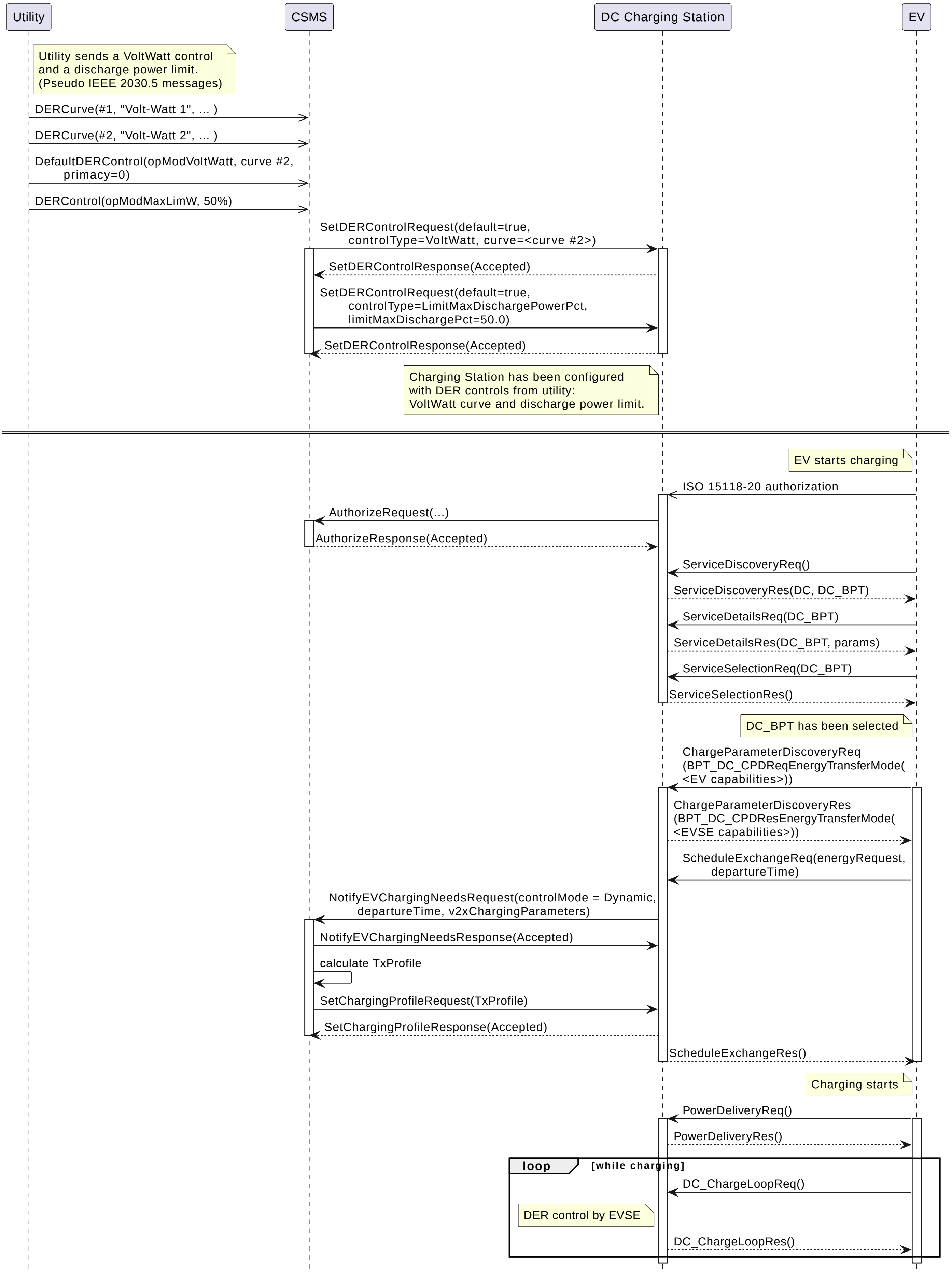

Utility
CSMS
DC Charging Station
EV
Utility sends a VoltWatt control and a discharge power limit. (Pseudo IEEE 2030.5 messages)
DERCurve(#1, "Volt-Watt 1", ...)
DERCurve(#2, "Volt-Watt 2", ...)
DefaultDERControl(opModVoltWatt, curve #2, primacy=0)
DERControl(opModMaxLimW, 50%)
SetDERControlRequest(default=true, controlType=VoltWatt, curve=<curve #2>)
SetDERControlResponse(Accepted)
SetDERControlRequest(default=true, controlType=LimitMaxDischargePowerPct, limitMaxDischargePct=50.0)
SetDERControlResponse(Accepted)
Charging Station has been configured with DER controls from utility: VoltWatt curve and discharge power limit.
EV starts charging
ISO 15118-20 authorization
AuthorizeRequest(...)
AuthorizeResponse(Accepted)
ServiceDiscoveryReq()
ServiceDiscoveryRes(DC, DC_BPT)
ServiceDetailsReq(DC_BPT)
ServiceDetailsRes(DC_BPT, params)
ServiceSelectionReq(DC_BPT)
ServiceSelectionRes()
DC_BPT has been selected
ChargeParameterDiscoveryReq (BPT_DC_CPDReqEnergyTransferMode(<EV capabilities>))
ChargeParameterDiscoveryRes (BPT_DC_CPDResEnergyTransferMode(<EVSE capabilities>))
ScheduleExchangeReq(energyRequest, departureTime)
NotifyEVChargingNeedsRequest(controlMode = Dynamic, departureTime, v2xChargingParameters)
NotifyEVChargingNeedsResponse(Accepted)
calculate TxProfile
SetChargingProfileRequest(TxProfile)
SetChargingProfileResponse(Accepted)
ScheduleExchangeRes()
Charging starts
PowerDeliveryReq()
PowerDeliveryRes()
loop
[while charging]
DC_ChargeLoopReq()
DER control by EVSE
DC_ChargeLoopRes()

Figure 207. Sequence diagram of an ISO 15118-20 DC bidirectional session with DER control in EVSE

R01 - Starting a V2X session with DER control in EVSE - Requirements

Table 161. R01 - Requirements

ID.	Precondition	Requirements	Note
R01.FR.01	Charging Station supports DC bidirectional power transfer	Charging Station SHALL expose the DER capabilities of the DC inverter, also referred to as nameplate information, in variables of the component DCDERCtrlr for each EVSE.	
R01.FR.02	When receiving a GetReportRequest for component DCDERCtrlr	A DC bidirectional Charging Station SHALL report all variables in DCDERCtrlr that are marked as mandatory.	This is to report the DER capabilities (or nameplate information) of each EVSE inverter.
R01.FR.03	Upon receiving DER controls as SetDERControlRequest messages from CSMS	Charging Station SHALL store all DER controls settings in the SetDERControlRequest persistently.	DER controls must persist after a power-cycle or reboot.

R02 - Starting a V2X session with DER control in EV

New in OCPP 2.1

No.	Type	Description
1	**Name**	Starting a V2X session with DER control in EV
2	**ID**	R02
3	**Objective(s)**	To start an AC bidirectional charging session with an EV that supports DER control of its inverter via ISO 15118-20.
4	**Description**	This use case illustrates how DER control can be executed in the scenarios where ISO 15118-20 AC_BPT_DER is used. A utility has set specific grid parameters (DER controls) that are required to be applied to bidirectional charging sessions. CSMS configures the settings at the Charging Station. An EV starts a bidirectional charging session using ISO 15118-20 AC_BPT_DER service. Charging Station configures DER control settings in EV, to be executed by the inverter in the EV.
	Actors	Utility, CSMS, Charging Station, EV

No.	Type	Description
	Scenario description	*An AC bidirectional session using AC_BPT_DER* 1. EV and Charging Station start an ISO 15118-20 session. 2. Upon successful authorization the EV starts service negotiation and requests service details fo the **AC_BPT_DER** service. 3. Charging Station presents service parameters that EV must comply with. a. Parameter DERControlFunctions in the service negotiation specifies in this use case that EV must at least support high frequency ride-through Must Trip mode (bit 11) and limitation of maximum discharge power (bit 20). 4. EV selects the AC_BPT_DER service. 5. EV and Charging Station exchange charge parameters. a. Charging Station sends EV a field DERControl with a FrequencyTrip curve and an ActivePowerSupport field containing the LimitMaxDischargePower with a PercentageValue of 50%. 6. Charging Station sends NotifyEVChargingNeedsRequest with *departureTime*, *v2xChargingParameters* and *derChargingParameters* that contains (among others) nameplate information from EV inverter and a list of DER controls supported by EV. 7. CSMS calculates a TxProfile for this session, which may contain periods of discharging, and sends it as a SetChargingProfileRequest. 8. Charging Station responds with ScheduleExchangeRes to EV. a. If *controlMode* = `Scheduled` is used, then ScheduleExchangeRes will contain the TxProfile schedule. b. If *controlMode* = `Dynamic` is used, then Charging Station will update charging/discharging levels via the ChargeLoop messages with EV to match the TxProfile schedule. 9. EV requests start of charging with a PowerDeliveryReq. 10. EV starts charging. 11. EV will start discharging according to schedule periods in TxProfile schedule. 12. DER control is executed by inverter in EV: a. It will stop discharging when the frequency deviates from nominal for a certain amount and time as defined in the high frequency trip curve. b. It will limit discharging power to 50% of its rated capacity as defined by the ActivePowerSupport limitation. c. When any of the above occurs EV will notify Charging Station about this via the DERAlarm element in the ChargeLoop message with *derFunctionName* set to `HFTrip` (a) or `LimitMaxPower` (b). 13. When a DER control influences the (dis)charging process, Charging Station will notify CSMS about this via the NotifyDERAlarmRequest with *controlType* set to `HFMustTrip` or `LimitDischargingPower` (depending on the event) and *gridEventFault* = `OverFrequency` in case of `HFMustTrip` and the *timestamp*.
5	**Prerequisites**	• In advance utility has sent DER control settings to the CSMS of CSO. In this example these are a VoltWatt curve and a limitation of discharging power to 50% of the rated power of the inverter. • CSMS has conveyed these settings as SetDERControlRequest messages to the Charging Station: 1. SetDERControlRequest(isDefault = true, controlType = `HVMustTrip`, curve = {…}) 2. SetDERControlRequest(isDefault = true, controlType = `LimitDischargePower`, limitMaxDischarge = { pctMaxDischargePower = 50.0}) See use cases R04 - Configure DER control settings at Charging Station.
6	**Post conditions**	
7	**Error Handling**	

No.	Type	Description
8	**Remarks**	The descriptions using ISO 15118 messages, such as the use of ChargeParameterDiscovery and ChargeLoop for DER control, are only informative and not prescriptive. The DER controls HFMustTrip and PowerMonitoring are just an example in this use case. In reality more DER controls are likely to be required.

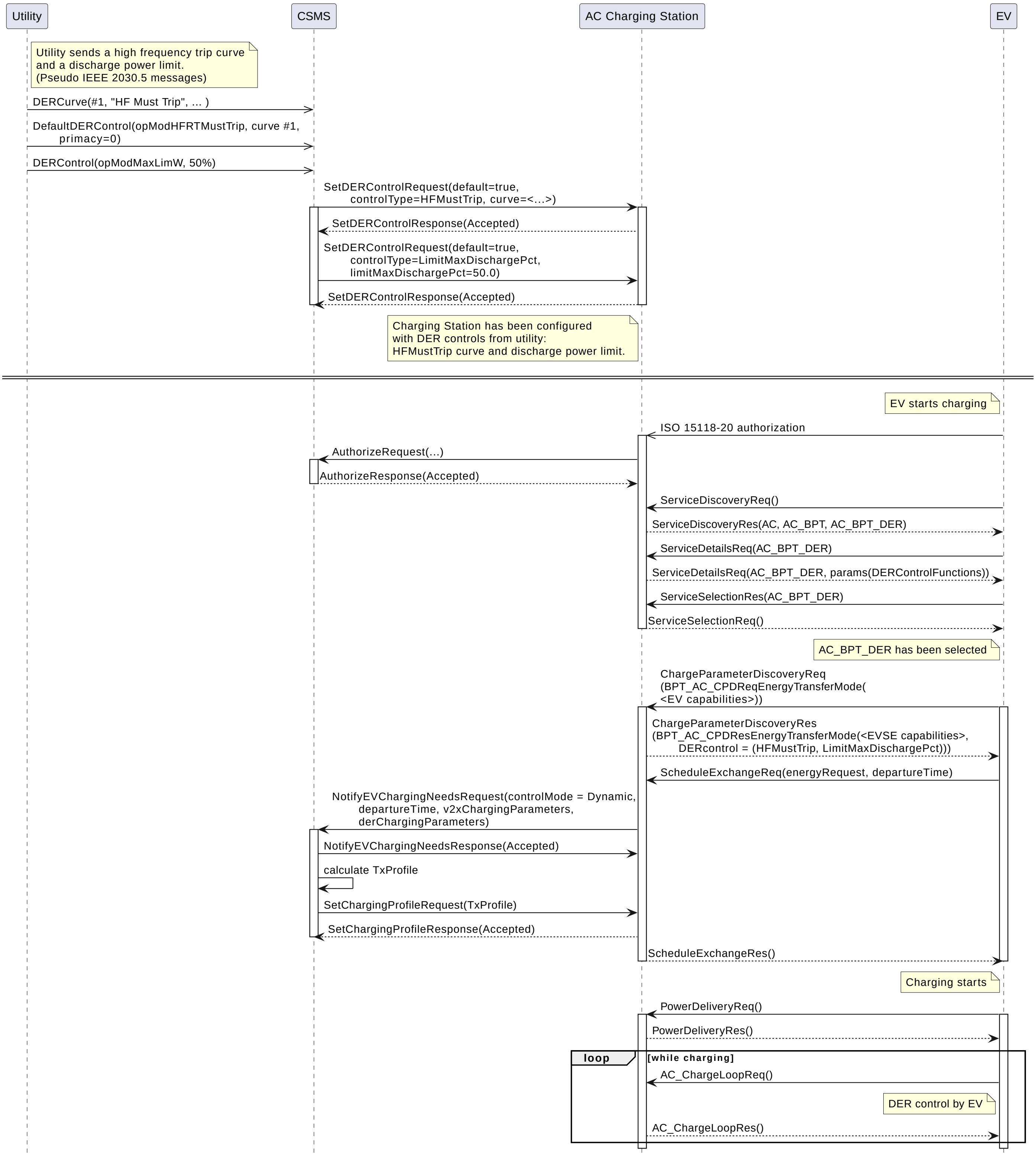

Utility
CSMS
AC Charging Station
EV
Utility sends a high frequency trip curve and a discharge power limit. (Pseudo IEEE 2030.5 messages)
DERCurve(#1, "HF Must Trip", ...)
DefaultDERControl(opModHFRTMustTrip, curve #1, primacy=0)
DERControl(opModMaxLimW, 50%)
SetDERControlRequest(default=true, controlType=HFMustTrip, curve=<...>)
SetDERControlResponse(Accepted)
SetDERControlRequest(default=true, controlType=LimitMaxDischargePct, limitMaxDischargePct=50.0)
SetDERControlResponse(Accepted)
Charging Station has been configured with DER controls from utility: HFMustTrip curve and discharge power limit.
EV starts charging
ISO 15118-20 authorization
AuthorizeRequest(...)
AuthorizeResponse(Accepted)
ServiceDiscoveryReq()
ServiceDiscoveryRes(AC, AC_BPT, AC_BPT_DER)
ServiceDetailsReq(AC_BPT_DER)
ServiceDetailsReq(AC_BPT_DER, params(DERControlFunctions))
ServiceSelectionRes(AC_BPT_DER)
ServiceSelectionReq()
AC_BPT_DER has been selected
ChargeParameterDiscoveryReq (BPT_AC_CPDReqEnergyTransferMode(<EV capabilities>))
ChargeParameterDiscoveryRes (BPT_AC_CPDResEnergyTransferMode(<EVSE capabilities>, DERcontrol = (HFMustTrip, LimitMaxDischargePct)))
ScheduleExchangeReq(energyRequest, departureTime)
NotifyEVChargingNeedsRequest(controlMode = Dynamic, departureTime, v2xChargingParameters, derChargingParameters)
NotifyEVChargingNeedsResponse(Accepted)
calculate TxProfile
SetChargingProfileRequest(TxProfile)
SetChargingProfileResponse(Accepted)
ScheduleExchangeRes()
Charging starts
PowerDeliveryReq()
PowerDeliveryRes()
loop
[while charging]
AC_ChargeLoopReq()
DER control by EV
AC_ChargeLoopRes()

Figure 208. Sequence diagram of an ISO 15118-20 AC bidirectional session with DER control in EV

R02 - Starting a V2X session with DER control in EV - Requirements

Table 162. R02 - Requirements

ID.	Precondition	Requirements	Note
R02.FR.01	Charging Station starts AC bidirectional power transfer session with EV via ISO 15118-20 AC_BPT_DER service	Charging Station SHALL expose the DER capabilities of the DC inverter in EV, also referred to as nameplate information, in the field *derChargeParameters* of a NotifyEVChargingNeedsRequest.	
R02.FR.02	Upon receiving DER controls as SetDERControlRequest messages from CSMS	Charging Station SHALL store all DER controls settings in the SetDERControlRequest persistently.	DER controls must persist after a power-cycle or reboot. (Same as R01.FR.03)
R02.FR.03	R02.FR.02	Charging Station SHALL send received DER controls to EV.	Using the ISO 15118-20 ChargeParameterDiscoveryRes message.

R03 - Starting a V2X session with hybrid DER control in both EV and EVSE

New in OCPP 2.1

No.	Type	Description
1	**Name**	Starting a V2X session with hybrid DER control in both EV and EVSE
2	**ID**	R03
3	**Objective(s)**	To start an AC bidirectional charging session with an EV that supports part of the DER control and an EVSE that supports another part.
4	**Description**	This use case illustrates how DER control can be executed in the scenario where ISO 15118-20 AC_BPT_DER is used, but not all controls are executed by the inverter in the EV. A utility has set specific grid parameters (DER controls) that are required to be applied to bidirectional charging sessions. CSMS configures the settings at the Charging Station. An EV starts a bidirectional charging session using ISO 15118-20 AC_BPT_DER service. Charging Station configures DER control settings in EV, to be executed by the inverter in the EV. DER controls that are not supported by EV are handled by EVSE.
	Actors	Utility, CSMS, Charging Station, EV

No.	Type	Description
	Scenario description	*A AC bidirectional session using AC_BPT_DER* 1. EV and Charging Station start an ISO 15118-20 session. 2. Upon successful authorization the EV starts service negotiation and requests service details fo the **AC_BPT_DER** service. 3. Charging Station presents service parameters that EV must comply with. a. Parameter DERControlFunctions in the service negotiation specifies in this use case that EV must at least support high frequency ride-through Must Trip mode (bit 11). 4. EV selects the AC_BPT_DER service. 5. EV and Charging Station exchange charge parameters. a. EV does not report support for `PowerMonitoring` in *evSupportedDERControl*. This can be emulated by the Charging Station if it is listed in the variable ACDERCtrlr.modesSupported. b. Charging Station sends a field DERControl with FrequencyTripCurve containing the FrequencyTrip curve. c. During charging the Charging Station will limit the maximum discharge power to 50% of the reported EVMaximumDischargePower in the ChargeLoopRes message with field EVSETargetActivePower. *Remaining steps 6 to 11 are identical to use case R02* 12. The following DER control is executed by inverter in EV: a. It will stop discharging when the frequency deviates from nominal for a certain amount and time as defined in the high frequency trip curve. b. When the above occurs EV will notify Charging Station about this via the DERAlarm element in the ChargeLoop message with *derFunctionName* set to `HFTrip`. 13. The following DER control is executed by Charging Station: a. It will limit discharging power to 50% of EVMaximumDischargePower as defined by the ActivePowerSupport limitation. 14. When a DER control influences the (dis)charging process, for example an overfrequency, Charging Station will notify CSMS about this via the NotifyDERAlarmRequest with *controlType* set to `HFMustTrip` and *gridEventFault* = `OverFrequency` and the *timestamp*.
5	**Prerequisites**	• In advance utility has sent DER control settings to the CSMS of CSO. In this example these are a high frequency trip curve (`HFMustTrip`) and a limitation of discharging power (`LimitMaxDischarge`) to 50% of the rated power of the inverter. • CSMS has conveyed the settings as SetDERControlRequest messages to Charging Station.
6	**Post conditions**	
7	**Error Handling**	
8	**Remarks**	The descriptions using ISO 15118 messages, such as the use of ChargeParameterDiscovery and ChargeLoop for DER control, are only informative and not prescriptive.

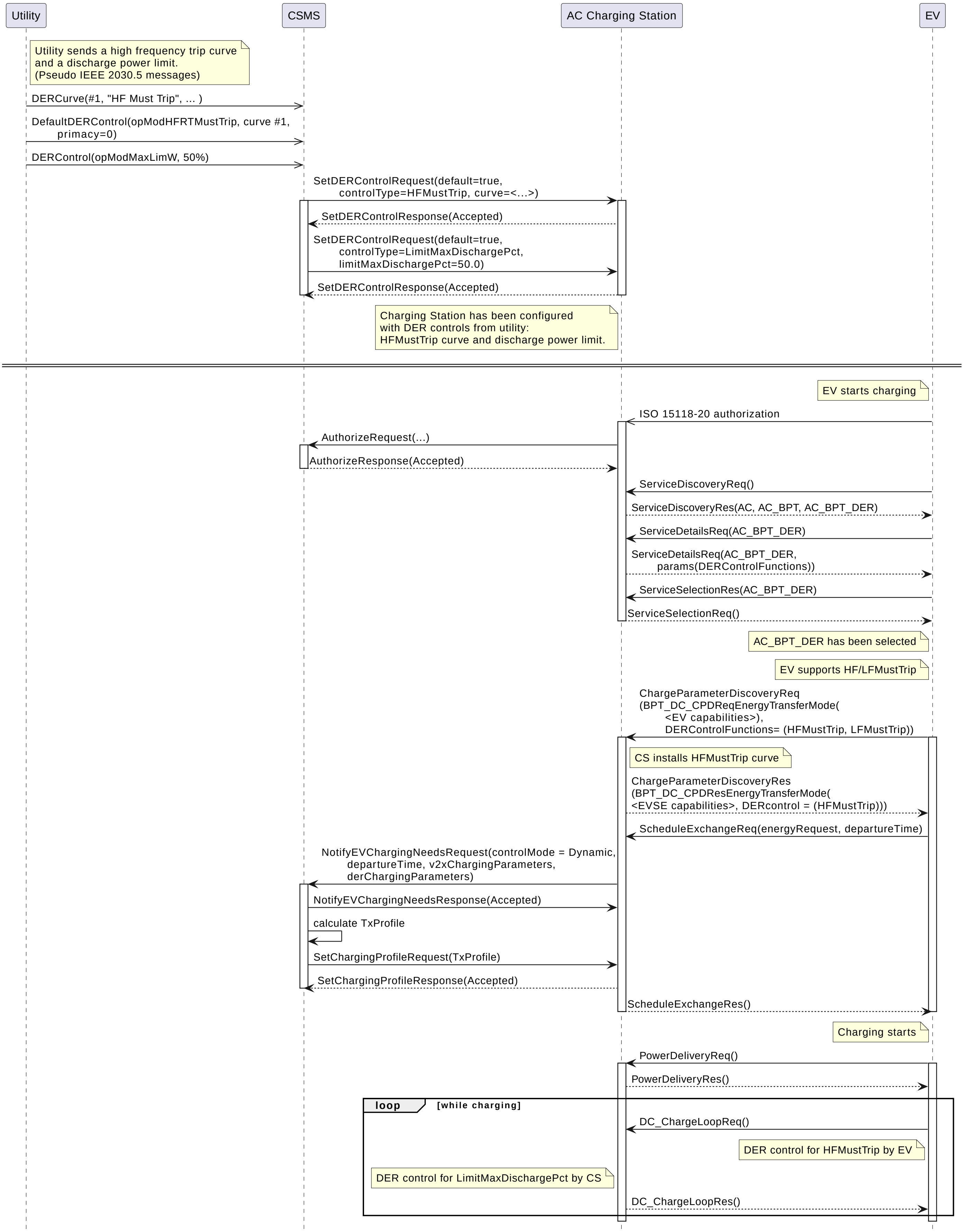

Utility
CSMS
AC Charging Station
EV
Utility sends a high frequency trip curve
and a discharge power limit.
(Pseudo IEEE 2030.5 messages)
DERCurve(#1, "HF Must Trip", ...)
DefaultDERControl(opModHFRTMustTrip, curve #1,
primacy=0)
DERControl(opModMaxLimW, 50%)
SetDERControlRequest(default=true,
controlType=HFMustTrip, curve=<...>)
SetDERControlResponse(Accepted)
SetDERControlRequest(default=true,
controlType=LimitMaxDischargePct,
limitMaxDischargePct=50.0)
SetDERControlResponse(Accepted)
Charging Station has been configured
with DER controls from utility:
HFMustTrip curve and discharge power limit.
EV starts charging
ISO 15118-20 authorization
AuthorizeRequest(...)
AuthorizeResponse(Accepted)
ServiceDiscoveryReq()
ServiceDiscoveryRes(AC, AC_BPT, AC_BPT_DER)
ServiceDetailsReq(AC_BPT_DER)
ServiceDetailsReq(AC_BPT_DER,
params(DERControlFunctions))
ServiceSelectionRes(AC_BPT_DER)
ServiceSelectionReq()
AC_BPT_DER has been selected
EV supports HF/LFMustTrip
ChargeParameterDiscoveryReq
(BPT_DC_CPDReqEnergyTransferMode(
<EV capabilities>),
DERControlFunctions= (HFMustTrip, LFMustTrip))
CS installs HFMustTrip curve
ChargeParameterDiscoveryRes
(BPT_DC_CPDResEnergyTransferMode(
<EVSE capabilities>, DERcontrol = (HFMustTrip)))
ScheduleExchangeReq(energyRequest, departureTime)
NotifyEVChargingNeedsRequest(controlMode = Dynamic,
departureTime, v2xChargingParameters,
derChargingParameters)
NotifyEVChargingNeedsResponse(Accepted)
calculate TxProfile
SetChargingProfileRequest(TxProfile)
SetChargingProfileResponse(Accepted)
ScheduleExchangeRes()
Charging starts
PowerDeliveryReq()
PowerDeliveryRes()
loop
[while charging]
DC_ChargeLoopReq()
DER control for HFMustTrip by EV
DER control for LimitMaxDischargePct by CS
DC_ChargeLoopRes()

Figure 209. Sequence diagram of an ISO 15118-20 AC bidirectional session with DER control in EV

R03 - Starting a V2X session with hybrid DER control in both EV and EVSE - Requirements

Table 163. R03 - Requirements

ID.	Precondition	Requirements	Note
R03.FR.01	Charging Station start AC bidirectional power transfer session with EV via ISO 15118-20 AC_BPT_DER service	Charging Station SHALL expose the DER capabilities of the DC inverter in EV, also referred to as nameplate information, in the field *derChargeParameters* of a NotifyEVChargingNeedsRequest.	(Same as R02.FR.01)
R03.FR.02	Upon receiving DER controls as SetDERControlRequest messages from CSMS	Charging Station SHALL store DER controls persistently.	DER controls must persist after a power-cycle or reboot. (Same as R01.FR.03)
R03.FR.03	If Charging Station is able to emulate certain DER controls locally	Charging Station SHALL list these DER controls in variable ACDERCtrlr.modesSupported	These are DER controls that Charging Station can emulate using ChargeLoop message to control the EV inverter.
R03.FR.04	If Charging Station is able to locally emulate a certain DER control	Charging Station MAY omit this DER control from the list or required DER controls in the ISO 15118-20 service negotiation for AC_BPT_DER.	This will allow an EV whose inverter does not support this DER control to participate in bidirectional power transfer, because this specific DER control is emulated by Charging Station.
R03.FR.05	If Charging Station is able to locally emulate a certain DER control, but this DER control is also supported by EV	Charging Station SHALL NOT emulate this DER control, but configure it in EV.	Capabilities of the inverter must be used much as possible.
R03.FR.06	R03.FR.02	Charging Station SHALL send received DER controls, that it does not emulate locally, to EV.	Using the ISO 15118-20 ChargeParameterDiscoveryRes message. (Same as R02.FR.02)

R04 - Configure DER control settings at Charging Station

New in OCPP 2.1

No.	Type	Description
1	**Name**	Configure DER control settings at Charging Station
2	**ID**	R04
3	**Objective(s)**	To show how a number of different DER settings can be configured.
4	**Description**	This use case contains several scenarios that show how DER settings are configured at a Charging Station. Not all possible DER controls are shown in these scenarios, but all are set as default or scheduled in a similar way.
	Actors	Utility, CSMS, Charging Station
	Scenario description #1	*Frequency droop as default DER control* 1. CSMS receives new frequency droop parameters from utility. 2. For each charging station in region of utility a. CSMS issues SetDERControlRequest messages with *isDefault* = true, *controlType* = `FreqDroop` and *freqDroop* field set to the frequency droop parameters: *overFreq*, *underFreq*, *overDroop*, *underDroop*, *responseTime* and **without** a *startTime* or *duration*. b. CS responds with SetDERControlResponse with *status* = `Accepted`. c. The default frequency droop will immediately become active.

No.	**Type**	**Description**
	Scenario description #2	*Frequency droop as a scheduled DER control* 1. CSMS receives new frequency droop parameters from utility. 2. For each charging station in region of utility a. CSMS issues SetDERControlRequest messages with *isDefault* = false, *controlType* = `FreqDroop` and *freqDroop* field set to the frequency droop parameters: *overFreq, underFreq, overDroop, underDroop, responseTime* and with a *startTime* and *duration*. b. CS responds with SetDERControlResponse with *controlType* = `FreqDroop` and *status* = `Accepted`. c. At *startTime* CS will report with a NotifyDERStartStopRequest that the frequency droop has become active (thus overruling any default setting for frequency droop, if present).
	Scenario description #3	*Ride-through curve as a default DER control* 1. CSMS receives new default HFRT curve from utility 2. For each charging station in region of utility a. CSMS installs new DER curve, using SetDERControlRequest with *controlType* = `HFMayTrip`, *isDefault* = true, and an object DERCurveType with *yUnit* = `Not_Applicable` and *curveData* containing the curve points. b. CS responds with SetDERControlResponse with *status* = `Accepted`. 3. If the net frequency deviates from nominal, only when it exceeds *curveData.x* Herz for more than *curveData.y* seconds, shall Charging Station stop exporting power to or importing power from the grid. a. When that happens Charging Station notifies CSMS via a NotifyDERAlarmRequest with *controlType* = `HFMayTrip` and *gridEventFault* = `OverFrequency`.
	Scenario description #4	*VoltVar Q(U) curve as a default DER control* 1. CSMS receives new scheduled VoltVar curve from utility 2. For each charging station in region of utility a. CSMS installs new DER curve, using SetDERControlRequest with *controlType* = `VoltVar` and a DERCurveType object with *yUnit* set to one of `PctMaxW`, `PctMaxVar` or `PctVarAvail` and *curveData* containing the curve points, and a *startTime* and *duration*. b. CS responds with SetDERControlResponse with *status* = `Accepted`. 3. At *startTime* CS will send a NotifyDERStartStopRequest with *controlId* for this curve and *started* = true. 4. From *startTime* onwards, for a period of *duration* seconds, Charging Station will continuously adjust the amount of reactive power according to the VoltVar curve by sending an EVSETargetReactivePower element without a corresponding EVSETargetActivePower element in the AC/DC_ChargeLoop messages. a. Since this does not affect the active power profile of the charging/discharging sessions, CS will not notify CSMS about it, **unless** the charging profile has a setting for *setpointReactive*, in which case CS sends a NotifyDERAlarmRequest with *controlType* = `VoltVar`. 5. At *startTime* + *duration* (i.e. when control ends) CS will send a NotifyDERStartStopRequest with *controlId* for the curve and *started* = false. **NOTE** According to ISO 15118-20, when in scheduled control mode, and the grid code only demands a specific reactive power behavior (e.g. resulting from a Q(U) rule), the SECC (EVSE) shall only send an EVSETargetReactivePower element without a corresponding EVSETargetActivePower element. This allows to satisfy grid code requirements without interfering in the normal EVPowerProfile.
5	**Prerequisites**	
6	**Post conditions**	
7	**Error Handling**	

No.	**Type**	**Description**
8	**Remarks**	This use case is similar for other DER controls. Charging station has to store these values persistently. The `HFMayTrip` curve configures ride-through behavior. The `HFMustTrip` curve configures the trip behavior.

R04 - Configure DER control settings at Charging Station - Requirements

Table 164. R04 - Requirements

ID.	Precondition	Requirements	Note
SetDERControl			
R04.FR.01	Charging Station receives a SetDERControlRequest with a *controlType* that it does not support	Charging Station SHALL respond with SetDERControlResponse with *status* = `NotSupported`.	
R04.FR.02	Charging Station receives a SetDERControlRequest with *isDefault* = true and a *controlType* that it supports AND no default DER control of *controlType* exists	Charging Station SHALL configure this DER control as the new default DER control, and respond with SetDERControlResponse with *status* = `Accepted`.	
R04.FR.03	Charging Station receives a SetDERControlRequest with *isDefault* = true and a *controlType* that it supports AND an existing default DER control of *controlType* has a higher value for *priority*	Charging Station SHALL set the *isSuperseded* attribute of the existing control to true and accept the new control as the new default DER control, and respond with SetDERControlResponse with *status* = `Accepted` and *supersededId* with Id of superseded control.	A lower *priority* value overrules a higher *priority* value.
R04.FR.04	(R04.FR.02 OR R04.FR.03) AND a scheduled DER control for *controlType* is already active	Charging Station SHALL accept this default DER control and respond with SetDERControlResponse with *status* = `Accepted` and continue to execute the scheduled DER curve.	Replacing a default curve while a scheduled curve is active (between *startTime* and *duration* seconds) will not affect the scheduled curve.
R04.FR.05	Charging Station receives a SetDERControlRequest with *isDefault* = false and a *controlType* that it supports AND no scheduled DER control of *controlType* exists	Charging Station SHALL accept this DER control, and respond with SetDERControlResponse with *status* = `Accepted`.	
R04.FR.06	Charging Station receives a SetDERControlRequest with *isDefault* = false and a *controlType* that it supports AND an existing scheduled DER control of *controlType*, that is not yet active, has a higher value for *priority*	Charging Station SHALL set the *isSuperseded* attribute of the existing control to true and accept the new control as the new scheduled DER control for *controlType*, and respond with SetDERControlResponse with *status* = `Accepted`, and *supersededId* with Id of superseded control.	A lower *priority* value has preference. The existing scheduled DER control is immediately superseded, even when *startTime* of the new control is later than that of the existing control.

R04.FR.07	Charging Station receives a SetDERControlRequest with a *controlType* that it supports AND an existing scheduled DER control of *controlType* is already active and has a higher value for *priority*	Charging Station SHALL respond with SetDERControlResponse with *status* = `Accepted` and SHALL continue to execute the existing control until *startTime* of the new control, upon which the *isSuperseded* attribute of the existing control is set to true and the new control is executed for *controlType*.	A lower *priority* value has preference. When the existing scheduled DER control is already active, it remains active until the *startTime* of the new control.
R04.FR.08	Charging Station receives a SetDERControlRequest with a *controlType* that it supports AND an existing scheduled DER control of *controlType* is already active and has a lower value for *priority*	Charging Station SHALL respond with SetDERControlResponse with *status* = `Accepted`, and *supersededId* = *controlId*, and SHALL set the *isSuperseded* attribute to true and continue to execute the existing setting for *controlId*.	A lower *priority* value has preference. This new control is immediately superseded by the existing control which has lower *priority* value. CSMS should avoid sending a DER control that has a higher *priority* value than what is already configured.
R04.FR.09		Charging Station SHALL NOT execute a control which has *isSuperseded* = true	
R04.FR.10		Charging Station SHALL delete a control after *startTime* + *duration* seconds has expired	
R04.FR.11	When there is no control active for *controlType* that has *isDefault* = false	Charging Station SHALL execute the control for *controlType* that has *isDefault* = true and *isSuperseded* = false	When there is no scheduled control active (after its *startTime*), then revert to the default control.
NotifyDERStartStop			
R04.FR.20	When a DER control has *isDefault* = false, *isSuperseded* = false and *startTime* equals current time	Charging Station SHALL send a NotifyDERStartStopRequest with *controlId* for this control, *started* = true, and *timestamp* is current time.	Start of DER control at *startTime* is reported.
R04.FR.21	(When a DER control A has *isDefault* = false, *isSuperseded* = false and *startTime* is current time) AND (another DER control B has *isDefault* = false, *isSuperseded* = false and a higher value for *priority* and is currently active)	Charging Station SHALL send a NotifyDERStartStopRequest with *controlId* = A, *started* = true, and *supersededId* = B and *timestamp* is current time.	Starting of A will supersede B.
R04.FR.22	When a DER control has *isDefault* = false, *isSuperseded* = false and *startTime* + *duration* equals current time	Charging Station SHALL send a NotifyDERStartStopRequest with *controlId* for this control, *started* = false, and *timestamp* is current time.	End of DER control after *duration* is reported.
GetDERControl			
R04.FR.30	Charging Station receives a GetDERControlRequest AND no DER controls are present on Charging Station for given *isDefault*, *controlType* and *controlId*	Charging Station SHALL return a *status* = `NotFound` in GetDERControlResponse.	A missing *isDefault*, *controlType* or *controlId* in the request matches any value of *isDefault*, *controlType* or *controlId*.
R04.FR.31	When Charging Station receives a GetDERControlRequest	The Charging Station SHALL set the *requestId* in every ReportDERControlRequest to the value of *requestId* in GetDERControlRequest.	

R04.FR.32	When the DER controls are reported in more than one ReportDERControlRequest	Charging Station SHALL set the *tbc* flag to **true** for all ReportDERControlRequest messages except the last.	
R04.FR.33	Charging Station receives a GetDERControlRequest without *isDefault*, *controlType* and *controlId* AND DER controls are present on Charging Station	Charging Station SHALL return a *status* = `Accepted` and send one or more ReportDERControlRequest messages for all controls.	Returns all controls.
R04.FR.34	NOT R04.FR.30 AND Charging Station receives a GetDERControlRequest with no *controlId* and with a *controlType* that it supports	Charging Station SHALL return a *status* = `Accepted` and send one or more ReportDERControlRequest messages for all controls that match the value of *controlType* and *isDefault* (if present) in the GetDERControlRequest .	Returns controls for *controlType* (matching *isDefault* if present).
R04.FR.35	NOT R04.FR.30 AND Charging Station receives a GetDERControlRequest with no *controlType* and with a *controlId*	Charging Station SHALL return a *status* = `Accepted` and send one or more ReportDERControlRequest messages for all controls that match the value of *controlId* and *isDefault* (if present).	Returns the controls for *controlId* (matching *isDefault* if present).
R04.FR.36	Charging Station receives a GetDERControlRequest with a *controlType* that it does not support	Charging Station SHALL return a *status* = `NotSupported` in GetDERControlResponse.	
ClearDERControl			
R04.FR.41	Charging Station receives a ClearDERControlRequest with no *controlId* and with a *controlType* that it supports, but that has not been set at the Charging Station for the specified value of *isDefault*	Charging Station returns a ClearDERControlResponse with *status* = `NotFound`.	
R04.FR.42	Charging Station receives a ClearDERControlRequest with no *controlType* and with a *controlId* that has not been set for the given value of *isDefault*	Charging Station SHALL respond with ClearDERControlResponse with *status* = `NotFound`.	
R04.FR.43	Charging Station receives a ClearDERControlRequest with no *controlId* and with a *controlType* that it does not support	Charging Station returns a ClearDERControlResponse with *status* = `NotSupported`.	
R04.FR.44	Charging Station receives a ClearDERControlRequest without *controlType* and without *controlId*	Charging Station SHALL clear all controls that match the value of *isDefault* in the request, and return a ClearDERControlResponse with *status* = `Accepted`.	Clear all default or all scheduled messages based on value of *isDefault*.

R04.FR.45	Charging Station receives a ClearDERControlRequest with no *controlId* and with a *controlType* that it supports and that is in use	Charging Station SHALL clear all controls that match the value of *isDefault* and *controlType* in the request, and return a ClearDERControlResponse with *status* = `Accepted`.	Return default or scheduled messages for *controlType* based on value of *isDefault*.

R05 - Charging station reporting a DER event

New in OCPP 2.1

No.	Type	Description
1	**Name**	Charging station reporting a DER event
2	**ID**	R05
3	**Objective(s)**	To show how a charging station reports that it is acting upon a VoltWatt DER curve, because this affects the power profile of the charging session.
4	**Description**	A grid anomaly occurs that forces a charging station to adjust behavior according to a preconfigured VoltWatt DER curve
	Actors	CSMS, Charging Station
	Scenario description	1. Voltage at grid connection point exceeds a threshold value from the VoltWatt curve while EV is discharging. 2. CS reports that it is adjusting discharging behavior according to VoltWatt curve via a NotifyDERAlarmRequest message with *controlType* = `VoltWatt` and *gridEventFault* = `OverVoltage`. 3. If CS is a DC station: CS lowers discharging power according to VoltWatt curve. 4. If CS is an AC station: CS instructs EV via ISO 15118-20 AC_ChargeLoop message to lower discharging power. CS repeats this every time the power value on the curve changes. 5. When voltage returns back to normal, CS reports end of this situation, via a NotifyDERAlarmRequest with *controlType* = `VoltWatt`, *gridEventFault* = `OverVoltage` and *alarmEnded* = true. 6. CS continues to follow the active charging profile.
5	**Prerequisites**	A VoltWatt curve has been configured at the charging station. In case of an AC station, the ISO 15118-20 AC_BPT service is used, or AC_BPT_DER is used without installing a VoltWatt curve in the EV, thus requiring the Charging Station to control power via the AC_ChargeLoop message. This assumes a discharging transaction was active during the entire event.
6	**Post conditions**	
7	**Error Handling**	
8	**Remarks**	

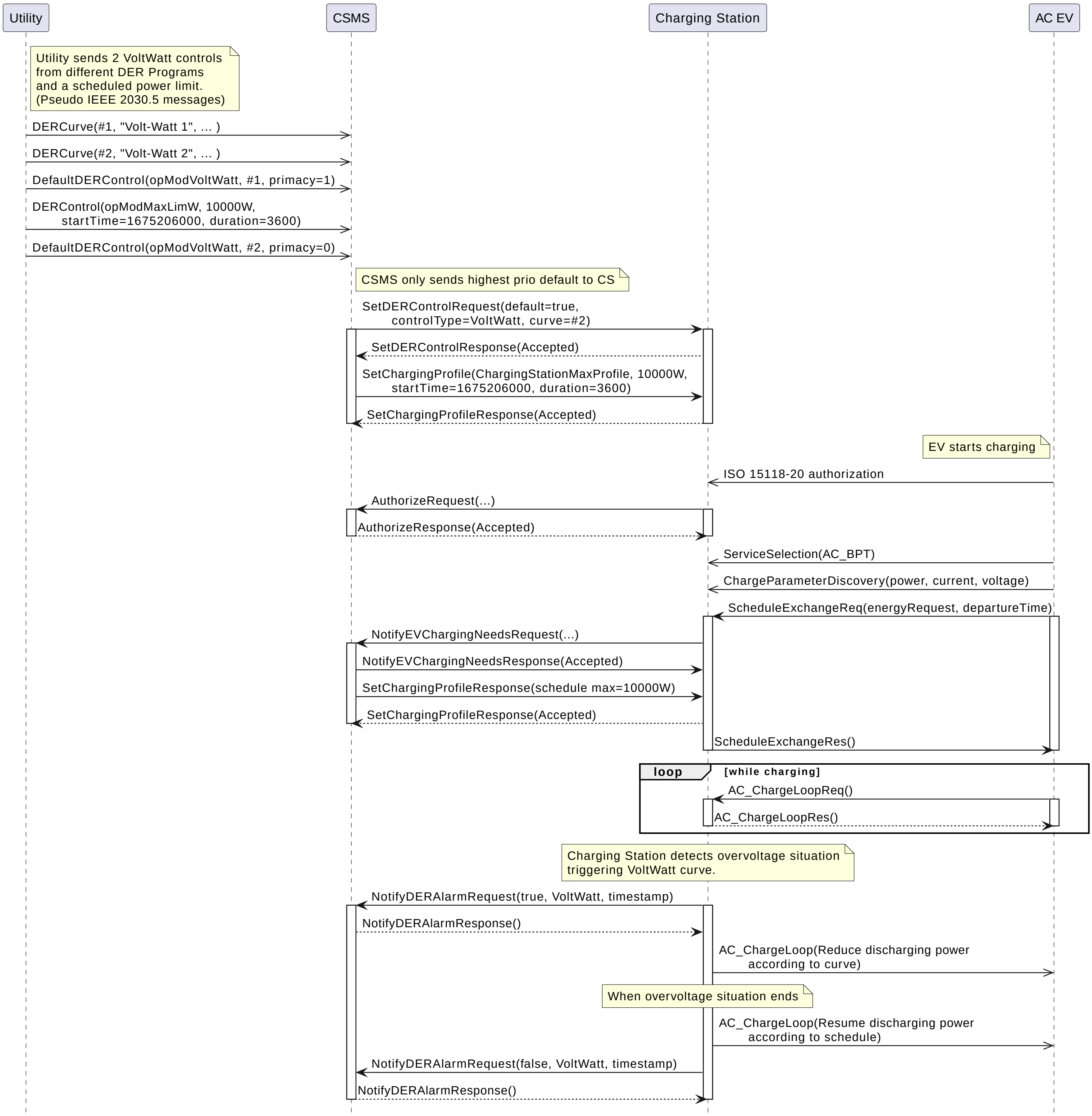

Utility
CSMS
Charging Station
AC EV
Utility sends 2 VoltWatt controls from different DER Programs and a scheduled power limit. (Pseudo IEEE 2030.5 messages)
DERCurve(#1, "Volt-Watt 1", ...)
DERCurve(#2, "Volt-Watt 2", ...)
DefaultDERControl(opModVoltWatt, #1, primacy=1)
DERControl(opModMaxLimW, 10000W, startTime=1675206000, duration=3600)
DefaultDERControl(opModVoltWatt, #2, primacy=0)
CSMS only sends highest prio default to CS
SetDERControlRequest(default=true, controlType=VoltWatt, curve=#2)
SetDERControlResponse(Accepted)
SetChargingProfile(ChargingStationMaxProfile, 10000W, startTime=1675206000, duration=3600)
SetChargingProfileResponse(Accepted)
EV starts charging
ISO 15118-20 authorization
AuthorizeRequest(...)
AuthorizeResponse(Accepted)
ServiceSelection(AC_BPT)
ChargeParameterDiscovery(power, current, voltage)
ScheduleExchangeReq(energyRequest, departureTime)
NotifyEVChargingNeedsRequest(...)
NotifyEVChargingNeedsResponse(Accepted)
SetChargingProfileResponse(schedule max=10000W)
SetChargingProfileResponse(Accepted)
ScheduleExchangeRes()
loop
[while charging]
AC_ChargeLoopReq()
AC_ChargeLoopRes()
Charging Station detects overvoltage situation triggering VoltWatt curve.
NotifyDERAlarmRequest(true, VoltWatt, timestamp)
NotifyDERAlarmResponse()
AC_ChargeLoop(Reduce discharging power according to curve)
When overvoltage situation ends
AC_ChargeLoop(Resume discharging power according to schedule)
NotifyDERAlarmRequest(false, VoltWatt, timestamp)
NotifyDERAlarmResponse()

Figure 210. Sequence diagram showing VoltWatt curve and charging limit from utility

R05 - Charging station reporting a DER event - Requirements

Table 165. R05 - Requirements

ID.	Precondition	Requirements	Note
R05.FR.01	When Charging Station needs to deviate from charging profile or nominal charging rate in case of no charging profile, because of a DER control setting or curve	Charging Station SHALL report this once per occurrence by sending a NotifyDERAlarmRequest with *timestamp* with current time and *controlType* that matches the DER control that causes this and *gridEventFault* for the fault that caused this.	

R05.FR.02	When Charging Station no longer needs to deviate from charging profile or nominal charging rate in case of no charging profile, because of a DER control setting or curve	Charging Station SHALL report the end of this situation by sending a NotifyDERAlarmRequest with *alarmEnded* = true, *timestamp* with current time and *controlType* that matches the DER control that is no longer taking over and *gridEventFault* for the fault that caused this.	
R05.FR.03	When a NotifyDERAlarmRequest with *alarmEnded* = false or absent has been reported for DER control A for *controlType* X and Charging Station needs to deviate from this because of a DER control B that is now activated for *controlType* Y	Charging Station SHALL report taking over by DER control B by sending a NotifyDERAlarmRequest with *alarmEnded* = false or absent, *timestamp* with current time and *controlType* = B and *gridEventFault* = Y.	
R05.FR.04	R05.FR.03	Charging Station SHALL NOT report the ending of the alarm by DER control A via NotifyDERAlarmRequest	There are no multiple levels of alarm. Alarm is still active. Only the *controlType* and *gridEventFault* have changed.

S. Battery Swapping

Chapter 1. Introduction

Battery swapping is the process whereby an EV driver exchanges the near empty EV battery for a charged EV battery. The returned battery is placed in a dock and then charged again. Once it reaches a certain level of state of charge it becomes eligible to be swapped again for an empty battery.

For light EVs, such as e-scooters, the swapping is a manual process performed by the EV driver. For regular EVs the swapping is an automated process that is initiated by the EV driver, but performed by a machine.

NOTE Depending on the type of EVs the battery swap can consist of a single battery or a set of batteries.

Although this section assumes that an EV driver returns a battery (or set of batteries) before receiving a new battery (or set of batteries), the reverse order is also supported by the described mechanism.

A battery swap action cannot be recorded by OCPP TransactionEvent messages. The action of swapping a battery is not considered a charging transaction; instead, it is a service for which the price can depend, for example, on the difference between the state of charge of the old and new batteries or be based on a monthly fee. OCPP 2.1 introduces a new message to record the swapping of batteries. The actual charging of batteries in the dock can still be tracked by CSMS via TransactionEvent messages.

A battery swap station has multiple slots that to hold batteries. In OCPP, a battery swap station is treated as a charging station. Similar to charging stations, where one EVSE charges one EV, OCPP assumes that, in a battery swap station, conceptually one EVSE powers one battery slot. (In practice, it may be possible that a single power supply unit is used to charge batteries in multiple slots one after the other). Different types of battery slots can be represented by specifying a different connector type for an EVSE in the Device Model variable ConnectorType. (See Example representation of a battery swap station in device model).

Chapter 2. Availability states for a battery slot

The OCPP availability states are about availability or occupancy of an EVSE/Connector. That means that:

- "Available" means no battery is present in slot.
- "Occupied" means a battery is present in slot.
- "Unavailable" means slot cannot be used.

It is important to understand that this differs from the "availability of the swapping service". An "Available" slot does not contain a battery for swapping, whereas an "Occupied" slot does have a battery that can be used for swapping (assuming it has a high enough state of charge).

Chapter 3. Use cases for Battery Swapping

S01 - Battery Swap Local Authorization

No.	Type	Description
1	**Name**	Battery Swap Local Authorization
2	**ID**	S01
3	**Objective(s)**	To show how a battery swap action is authorized locally at the swap station.
4	**Description**	EV Driver authorizes locally at the swap station by presenting an RFID card.
	Actors	Charging Station, CSMS
	Scenario description	1. EV Driver authorizes at swap station by presenting a valid RFID card. 2. Charging Station (swap station) sends an AuthorizeRequest with *idToken* to CSMS. a. CSMS grants authorization to swap in the AuthorizeResponse with *idTokenInfo.status* = `Accepted`. 3. Charging Station opens or indicates empty slot(s) where the set of empty batteries can be inserted.
5	**Prerequisite(s)**	EV Driver has a valid contract for swapping.
6	**Postcondition(s)**	When successful, use case S03 - Battery Swap In/Out starts.
7	**Error handling**	When Charging Station has not enough batteries available for swapping, the swap action is refused and use case ends.
8	**Remark(s)**	Authorization of EV Driver using an RFID card follows the process of use case C01 - EV Driver Authorization using RFID. Since the authorization for battery swapping is not used to start a transaction, any requirements in C01 that are about behavior of authorization when a transaction for the idToken is in progress, are not relevant, because the situation does not occur.

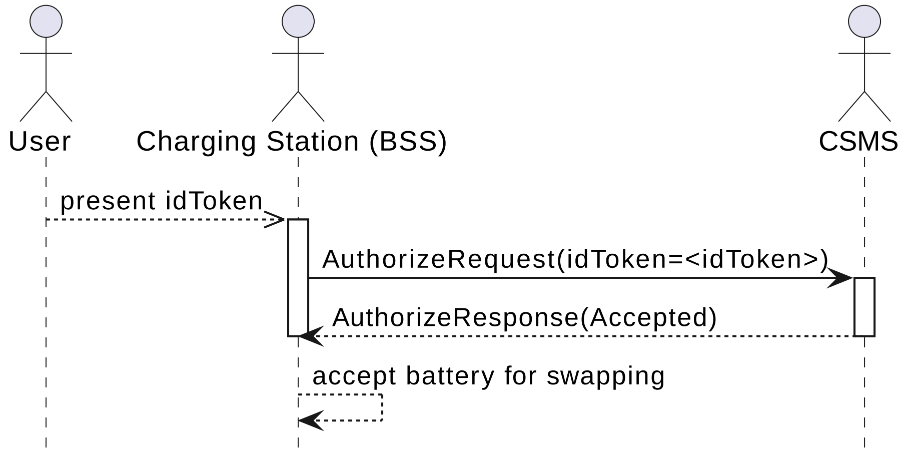

User
Charging Station (BSS)
CSMS
present idToken
AuthorizeRequest(idToken=<idToken>)
AuthorizeResponse(Accepted)
accept battery for swapping

Figure 211. Sequence diagram for local authorization

S01 - Battery Swap Local Authorization - Requirements

ID	Precondition	Requirement definition	Note
S01.FR.01	Upon successful authorization of AuthorizeRequest for an *idToken* AND when enough batteries are available for a swap	Charging Station (swap station) SHALL open or indicate an empty slot(s) where returned batteries can be inserted.	
S01.FR.02	When an *idToken* has been authorized and the EV Driver does not insert a battery into a slot before the timeout set by the Configuration Variable: `BatterySwapInTimeout`	Charging Station SHALL end the authorization of *idToken* and abort the battery swap process.	At this point no BatterySwapRequest has been sent yet.

S02 - Battery Swap Remote Start

No.	Type	Description
1	**Name**	Battery Swap Remote Start
2	**ID**	S02
3	**Objective(s)**	To show how a battery swap action is requested remotely by CSMS.
4	**Description**	EV Driver requests CSMS to initiate a battery swap via a smartphone app, e.g. by scanning a QR code or selecting the appropriate station in the app.
	Actors	Charging Station, CSMS
	Scenario description	1. When CSMS authorizes the EV Driver's request to charge via a smartphone app, CSMS sends a RequestBatterySwapRequest with *idToken* set to the authorization token for the EV Driver and a *requestId* to be used in the following BatterySwapRequest. 2. When Charging Station has at least as many batteries available for swapping as needed for the swap: a. Charging Station responds with RequestBatterySwapResponse with *status* = `Accepted`.
5	**Prerequisite(s)**	*idToken* is authorized to swap.
6	**Postcondition(s)**	When successful, use case S03 - Battery Swap In/Out follows.
7	**Error handling**	If not enough batteries are available for swapping, the Charging Station responds with RequestBatterySwapResponse with *status* = `Rejected` and *statusInfo.reasonCode* = "NoBatteryAvailable".
8	**Remark(s)**	

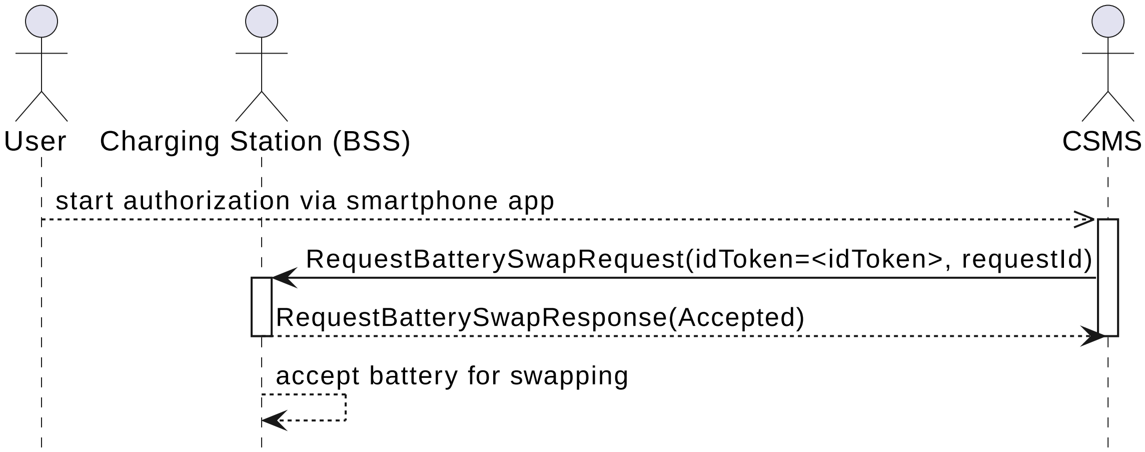

User
Charging Station (BSS)
CSMS
start authorization via smartphone app
RequestBatterySwapRequest(idToken=<idToken>, requestId)
RequestBatterySwapResponse(Accepted)
accept battery for swapping

Figure 212. Sequence diagram for remote start

S02 - Battery Remote Start - Requirements

ID	Precondition	Requirement definition	Note
S02.FR.01	When CSMS sends a RequestBatterySwapRequest for *idToken* with a *requestId* AND *idToken* is authorized by CSMS AND Charging Station has enough batteries available for swapping	Charging Station SHALL respond with RequestBatterySwapResponse with *status* = `Accepted`	
S02.FR.02	S02.FR.01	Charging Station SHALL use the same *requestId* in the BatterySwapRequest that results from this RequestBatterySwapRequest	This allows CSMS to associate the BatterySwapRequest with the RequestBatterySwapRequest.
S02.FR.03		CSMS SHALL NOT send send a RequestBatterySwapRequest for an *idToken* that is not authorized	

S02.FR.04	When CSMS sends a RequestBatterySwapRequest for *idToken* AND *idToken* is authorized by CSMS AND Charging Station has not enough batteries available for swapping	Charging Station SHALL respond with RequestBatterySwapResponse with *status* = `Rejected` and *statusInfo.reasonCode* = "NoBatteryAvailable"	More information can be provided in field *additionalInfo*.
S02.FR.05	S02.FR.01 AND the EV Driver does not insert a battery into a slot before the timeout set by the Configuration Variable: `BatterySwapInTimeout`	Charging Station SHALL end the authorization of *idToken* and abort the battery swap process.	At this point no BatterySwapRequest has been sent yet.

S03 - Battery Swap In/Out

No.	Type	Description
1	**Name**	Battery Swap In/Out
2	**ID**	S03
3	**Objective(s)**	To show how the action of returning the empty battery and accepting a charge battery is recorded.
4	**Description**	*This use case refers to a "set of batteries", but the set can consist of only a single battery.* An EV driver presents an empty battery or set of batteries to be swapped. The battery swap station accepts the set of empty batteries and returns set of charged batteries. The swap station records parameters, like serial number, state of charge and state of health of the swapped batteries.
	Actors	Charging Station, CSMS
	Scenario description	1. Charging Station opens or indicates the slot(s) where the set of empty batteries can be inserted. 2. EV Driver (or swapping machinery) inserts set of empty batteries into empty slot(s) of swap station. 3. Charging Station sends a BatterySwapRequest with *eventType* = `BatteryIn` and a *requestId* and with *idToken* set to the authorized *idToken*, and a *batteryData* field for each returned battery with *evseId* set to number of the slot where the battery was inserted, and the parameters *serialNumber*, *SoC*, *SoH*. 4. If CSMS accepts this battery, then a. CSMS responds with BatterySwapResponse with *status* = `Accepted`. b. Charging Station sends a NotifyEventRequest for the EVSE(s) corresponding to the used slot(s) with variable "AvailabilityState" set to "Occupied". 5. If CSMS does not accept this set of batteries, for example, because one is damaged, then a. CSMS responds with BatterySwapResponse with *status* = `Rejected` and *statusInfo.reasonCode* = "BatteryDamaged". (When desired, more specific information can be given in *additionalInfo*).
		6. Charging Station returns a set of batteries to EV Driver to use. *(The actual method or returning a battery is up to swap station implemention. A battery can be presented by station or user can select one.)* 7. If a battery was still charging a. Charging Station ends the charging transaction and sends a TransactionEventRequest with *eventType* = `Ended`. 8. EV Driver (or swapping machinery) accepts set of batteries and installs it in EV. 9. Charging Station sends a BatterySwapRequest with *eventType* = `BatteryOut` and the *requestId* from the `BatteryIn` event and a *batteryData* field with *evse* set to number of the slot where each battery is taken from, and the parameters *serialNumber*, *SoC*, *SoH*. a. CSMS responds with BatterySwapResponse with *status* = `Accepted`. b. Charging Station sends a NotifyEventRequest for the EVSE corresponding to the slot with variable "AvailabilityState" set to "Available". 10. EV Driver leaves.
5	**Prerequisite(s)**	

No.	Type	Description
6	**Postcondition(s)**	When successful, use case S04 - Battery Swap Charging follows.
7	**Error handling**	If CSMS does not accept the inserted battery, it will respond with a BatterySwapResponse with *status* = `Rejected` and *statusInfo* with a *reasonCode*. Predefined reason codes contain: BatterySoHLow, BatterySoC, BatteryDamaged, BatteryUnknown, BatteryType, NoBatteryAvailable. See Appendix 5 Reason Codes.
8	**Remark(s)**	A battery swap station is treated as a Charging Station in OCPP in which every battery slot represents a logical EVSE. The BatterySwapRequest message is related to the *idToken* of the EV Driver. The TransactionEvent messages are not associated with the EV Driver, hence the *idToken* in the TransactionEvent can be the predefined value in `BatterySwapIdtoken` or be absent. Battery swap stations usually operate as "old battery in first, new battery out second", as described in this use case. The described mechanism also works, however, when the order is reversed. The use of a StatusNotificationRequest instead of a NotifyEventRequest to report AvailabilityState, is allowed, but use of StatusNotificationRequest has been deprecated.

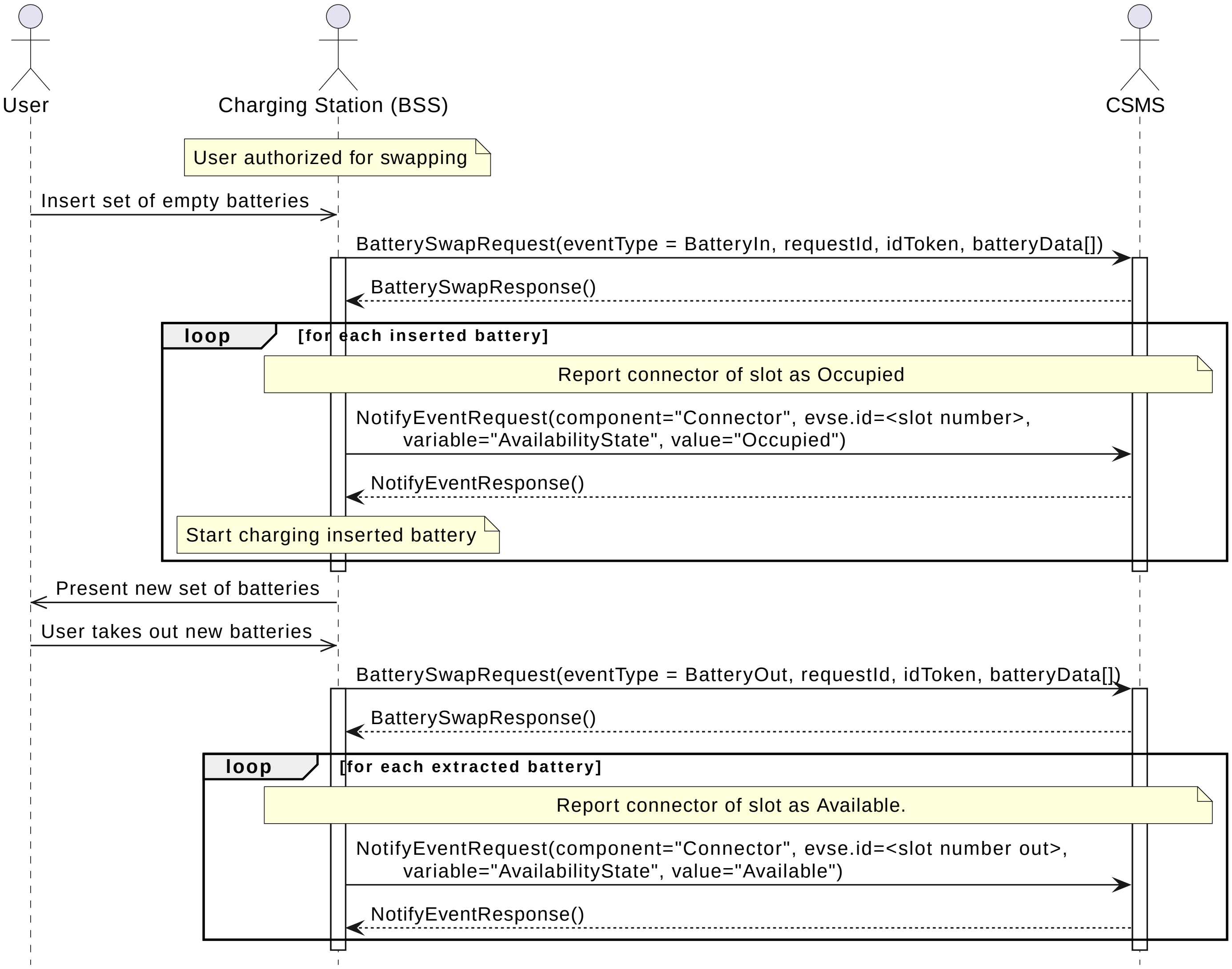

User
Charging Station (BSS)
CSMS
User authorized for swapping
Insert set of empty batteries
BatterySwapRequest(eventType = BatteryIn, requestId, idToken, batteryData[])
BatterySwapResponse()
loop
[for each inserted battery]
Report connector of slot as Occupied
NotifyEventRequest(component="Connector", evse.id=<slot number>, variable="AvailabilityState", value="Occupied")
NotifyEventResponse()
Start charging inserted battery
Present new set of batteries
User takes out new batteries
BatterySwapRequest(eventType = BatteryOut, requestId, idToken, batteryData[])
BatterySwapResponse()
loop
[for each extracted battery]
Report connector of slot as Available.
NotifyEventRequest(component="Connector", evse.id=<slot number out>, variable="AvailabilityState", value="Available")
NotifyEventResponse()

Figure 213. Sequence diagram for swapping batteries

S03 - Battery Swap In/Out - Requirements

ID	Precondition	Requirement definition	Note
S03.FR.01	When a set of batteries is inserted into a slot	Charging Station SHALL send a BatterySwapRequest with *eventType* = `BatteryIn`, a *requestId*, an *idToken* set to authorized *idToken*, and *batteryData* with parameters from each battery with *evse* set to number of the slot where battery is inserted	*requestId* is used to associate BatteryIn and BatteryOut events in CSMS. A set can also consist of a single battery.
S03.FR.02	S03.FR.01	Charging Station SHALL report it as a Connector that is `Occupied` by a NotifyEventRequest for variable "AvailabilityState" of the Connector.	
S03.FR.03	When a set of batteries is removed from a slot	Charging Station SHALL send a BatterySwapRequest with *eventType* = `BatteryOut`, the same *requestId* as used during `BatteryIn`, an *idToken* set to authorized *idToken*, and *batteryData* with parameters from the battery with *evse* set to number of the slot where battery is taken from.	
S03.FR.04	S03.FR.03	Charging Station SHALL report it as a Connector that is `Available` by a NotifyEventRequest for variable "AvailabilityState" of the Connector.	
S03.FR.05	When EV Driver authorizes to perform a battery swap AND Charging has no (or not enough) batteries available for swapping	Charging Station SHALL refuse the battery swap operation.	
S03.FR.06	When the set of batteries offered after insertion of the old batteries is not removed after the timeout specified in `BatterySwapOutTimeout`	Charging Station SHALL send a BatterySwapRequest with *eventType* = `BatteryOutTimeout`, the same *requestId* as used during `BatteryIn` and *idToken* set to the authorized *idToken*, and *batteryData* with parameters from the battery with *evse* set to the number of the slot from which the battery was offered to EV Driver.	Situation needs to be reported, because CSMS ends up with an orphan `BatteryIn` for which a `BatteryOut` is missing.

S04 - Battery Swap Charging

No.	Type	Description
1	**Name**	Battery Swap Charging
2	**ID**	S04
3	**Objective(s)**	To show how the charging of swapped batteries is recorded.
4	**Description**	*This use case refers to a "set of batteries", but the set can consist of only a single battery.* An EV driver has swapped empty batteries for new ones, which the Charging Station will charge again up to a predefined state of charge.
	Actors	Charging Station, CSMS
	Scenario description	1. Charging Station charges the empty batteries in the swap station. a. Charging Station sends a TransactionEventRequest with *eventType* = `Started` and *evse* set to number of the slot where the battery is in, *idToken* set to a predefined value in `BatterySwapIdtoken` or left empty with *type* = `NoAuthorization`, and *triggerReason* = `CablePluggedIn`. *(`CablePluggedIn` for a swap station means that the battery has been inserted.)* b. This follows the regular charging process as described in use cases E02 or E03. 2. While charging the battery a. Charging Station periodically sends a TransactionEventRequest with the current state of charge of the battery as a *metervalue* with *measurand* = "SoC" 3. When a battery has been charged to the maximum allowed state of charge: a. Charging Station suspends the charging transaction and sends a TransactionEventRequest with *eventType* = `Updated` and *chargingState* = `SuspendedEVSE` and *triggerReason* = `EnergyLimitReached`.
5	**Prerequisite(s)**	A battery is present to be charged.

No.	Type	Description
6	**Postcondition(s)**	Battery is charged until `BatterySwapMaxSoc`.
7	**Error handling**	
8	**Remark(s)**	A battery swap station is treated as a Charging Station in OCPP in which every battery slot represents a logical EVSE. The BatterySwap message is related to the *idToken* of the EV Driver. The TransactionEvent messages are not associated with the EV Driver, hence the *idToken* in the TransactionEvent can be the predefined value in `BatterySwapIdtoken` or be absent. The battery swap station may suspend and resume the charging of batteries or change charging rate as needed, or CSMS can influence the charging rate. This is analogous to local load-balancing and smart charging in regular charging stations.

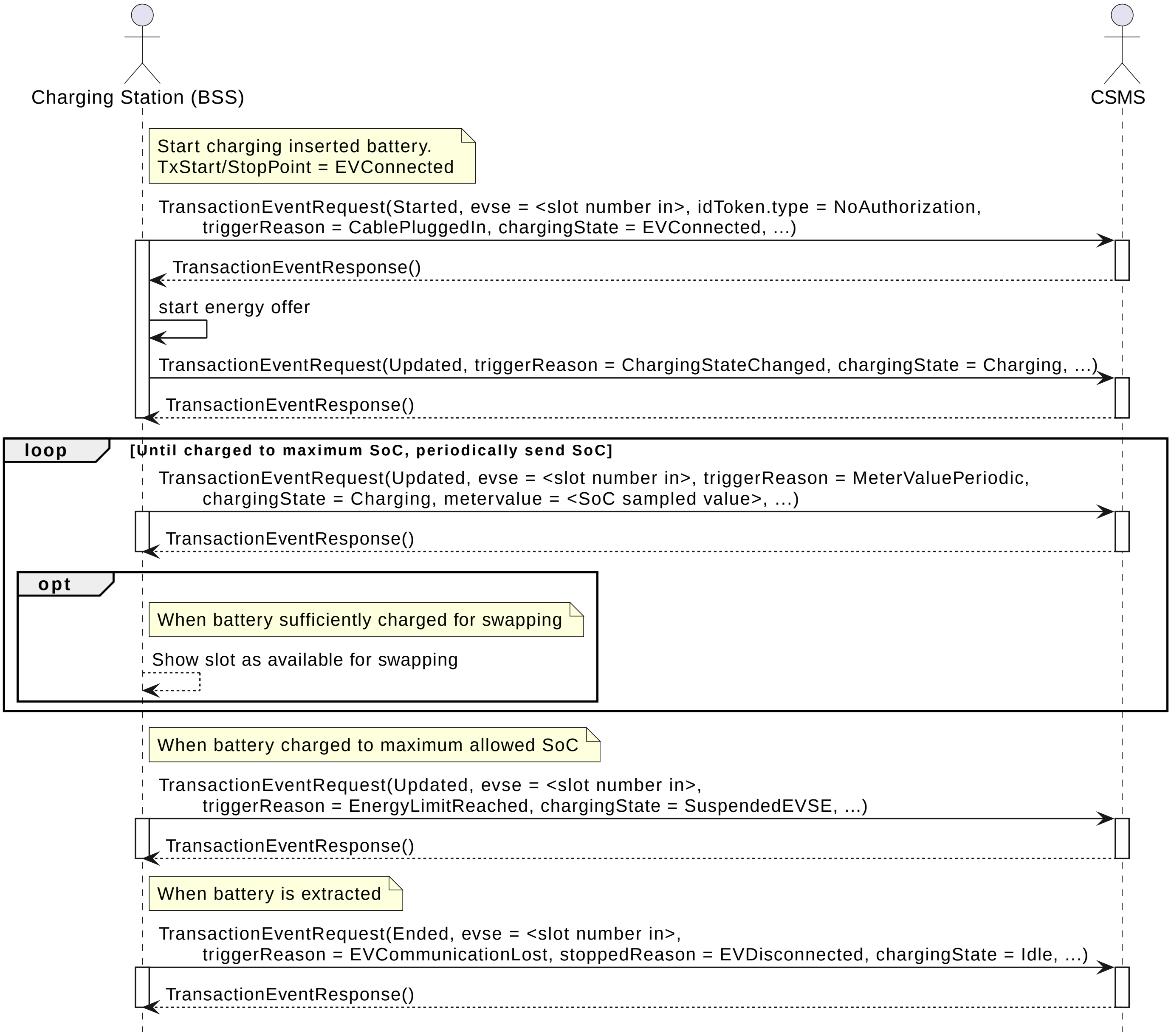

Charging Station (BSS)
CSMS
Start charging inserted battery.
TxStart/StopPoint = EVConnected
TransactionEventRequest(Started, evse = <slot number in>, idToken.type = NoAuthorization,
triggerReason = CablePluggedIn, chargingState = EVConnected, ...)
TransactionEventResponse()
start energy offer
TransactionEventRequest(Updated, triggerReason = ChargingStateChanged, chargingState = Charging, ...)
TransactionEventResponse()
loop
[Until charged to maximum SoC, periodically send SoC]
TransactionEventRequest(Updated, evse = <slot number in>, triggerReason = MeterValuePeriodic,
chargingState = Charging, metervalue = <SoC sampled value>, ...)
TransactionEventResponse()
opt
When battery sufficiently charged for swapping
Show slot as available for swapping
When battery charged to maximum allowed SoC
TransactionEventRequest(Updated, evse = <slot number in>,
triggerReason = EnergyLimitReached, chargingState = SuspendedEVSE, ...)
TransactionEventResponse()
When battery is extracted
TransactionEventRequest(Ended, evse = <slot number in>,
triggerReason = EVCommunicationLost, stoppedReason = EVDisconnected, chargingState = Idle, ...)
TransactionEventResponse()

Figure 214. Sequence diagram for charging swapped batteries

S04 - Battery Swap Charging - Requirements

ID	Precondition	Requirement definition	Note
S04.FR.01	When a battery is inserted into a slot	Charging Station SHALL start charging the battery.	

ID	Precondition	Requirement definition	Note
S04.FR.02	S04.FR.01 AND `BatterySwapIdtoken` is set	Charging Station SHALL send a TransactionEventRequest with *eventType* = `Started` and *triggerReason* = `CablePluggedIn` and *chargingState* = `Charging` and *idToken.idToken* = `BatterySwapIdtoken` and *idToken.type* = `Central`.	
S04.FR.03	S04.FR.01 AND `BatterySwapIdtoken` is not set	Charging Station SHALL send a TransactionEventRequest with *eventType* = `Started` and *triggerReason* = `CablePluggedIn` and *chargingState* = `Charging` and *idToken.idToken* = "" and *idToken.type* = `NoAuthorization`.	
S04.FR.04		Charging Station SHALL send TransactionEventRequest messages with *eventType* = `Updated` with periodic meter values for *measurand* = "SoC" to inform CSMS of the state of charge of the battery.	This is to inform CSMS about progress of charging. See `SampledDataTxUpdatedMeasurands`. The *triggerReason* will be `MeterValuePeriodic` if there is no other reason to send a TransactionEventRequest.
S04.FR.05	When a battery in a slot is charged to `BatterySwapTargetSoc`	Charging Station SHALL make the battery in this slot eligible for swapping.	
S04.FR.06	When a battery in a slot is charged to `BatterySwapMaxSoc`	Charging Station SHALL suspend energy transfer and send a TransactionEventRequest with *eventType* = `Updated` and *triggerReason* = `EnergyLimitReached` and *chargingState* = `SuspendedEVSE`.	Transaction is kept alive to allow that energy transfer is started again (e.g. in case of V2X operations)
S04.FR.07	When a battery is removed from a slot	Charging Station SHALL send a TransactionEventRequest with *eventType* = `Ended` and *triggerReason* = `EVCommunicationLost` and *chargingState* = `Idle` and *stoppedReason* = `EVDisconnected`.	
S04.FR.08		Configuration key TxStartPoint SHALL be "EVConnected"	The transaction is started when the battery is inserted.
S04.FR.09		Configuration key TxStopPoint SHALL be "EVConnected"	The transaction is ended when the battery is removed.
S04.FR.10		`BatterySwapMaxSoc` SHALL be greater than or equal to `BatterySwapTargetSoc`.	
S04.FR.11	When Charging Station reboots	Charging Station SHALL start new transactions for all EVSEs that have a battery in their slot.	This includes the ones that have reached their maximum state of charge (`BatterySwapMaxSoc`). See also S04.FR.06.

Messages, Datatypes & Enumerations

Chapter 1. Messages

1.1. AdjustPeriodicEventStream

1.1.1. AdjustPeriodicEventStreamRequest

Class

Field Name	Field Type	Card.	Description
id	integer, 0 <= val	1..1	Required.
params	PeriodicEventStreamParamsType	1..1	Required. Updated rate of sending data

1.1.2. AdjustPeriodicEventStreamResponse

Class

Field Name	Field Type	Card.	Description
status	GenericStatusEnumType	1..1	Required. Status of operation.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information

1.2. AFRRSignal

1.2.1. AFRRSignalRequest

(2.1) This message passes an aFRR signal on to the charging station. Charging station uses the value of *signal* to select a matching *power* value from the *v2xSignalWattCurve* in the ChargingSchedulePeriod.

Class

Field Name	Field Type	Card.	Description
timestamp	dateTime	1..1	Required. Time when signal becomes active.
signal	integer	1..1	Required. Value of signal in *v2xSignalWattCurve*.

1.2.2. AFRRSignalResponse

(2.1) Response stating whether signal was accepted. Response will be `Accepted` if a *v2xSignalWattCurve_* element exists in the ChargingSchedulePeriodType for that point in time.

Class

Field Name	Field Type	Card.	Description
status	GenericStatusEnumType	1..1	Required.
statusInfo	StatusInfoType	0..1	Optional. Additional information on status.

1.3. Authorize

1.3.1. 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..10000]	0..1	Optional. **(2.1)** 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.

1.3.2. 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'.
allowedEnergyTransfer	EnergyTransferModeEnumType	0..*	Optional. **(2.1)** List of allowed energy transfer modes the EV can choose from. If omitted this defaults to charging only.
idTokenInfo	IdTokenInfoType	1..1	Required. This contains information about authorization status, expiry and group id.
tariff	TariffType	0..1	Optional. **(2.1)** Tariff for this *idToken*.

1.4. BatterySwap

1.4.1. BatterySwapRequest

(2.1) Message sent by Charging Station when a battery is swapped in or out of a battery swap station.

Class

Field Name	Field Type	Card.	Description
eventType	BatterySwapEventEnumType	1..1	Required. Battery in/out
requestId	integer	1..1	Required. RequestId to correlate BatteryIn/Out events and optional RequestBatterySwapRequest.
idToken	IdTokenType	1..1	Required. Id token of EV Driver
batteryData	BatteryDataType	1..*	Required. Info on batteries inserted or taken out.

1.4.2. BatterySwapResponse

(2.1) Empty response by CSMS to confirm receipt of BatterySwapRequest.

1.5. BootNotification

1.5.1. 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

1.5.2. 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.
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.

1.6. CancelReservation

1.6.1. 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, 0 ⇐ val	1..1	Required. Id of the reservation to cancel.

1.6.2. 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.

1.7. CertificateSigned

1.7.1. 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 typeOfCertificate was included in the SignCertificateRequest that requested this certificate to be signed AND both the 15118 connection and the Charging Station connection are implemented.
requestId	integer	0..1	Optional. **(2.1)** RequestId to correlate this message with the SignCertificateRequest.

1.7.2. 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.

1.8. ChangeAvailability

1.8.1. 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.

1.8.2. 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.

1.9. ChangeTransactionTariff

1.9.1. ChangeTransactionTariffRequest

Class

Field Name	Field Type	Card.	Description
transactionId	identifierString[0..36]	1..1	Required. Transaction id for new tariff.
tariff	TariffType	1..1	Required. New tariff to use for transaction.

1.9.2. ChangeTransactionTariffResponse

Class

Field Name	Field Type	Card.	Description
status	TariffChangeStatusEnumType	1..1	Required. Status of the operation
statusInfo	StatusInfoType	0..1	Optional. Detailed status information

1.10. ClearCache

1.10.1. ClearCacheRequest

This contains the field definition of the ClearCacheRequest PDU sent by the CSMS to the Charging Station. No fields are defined.

1.10.2. 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.

1.11. ClearChargingProfile

1.11.1. 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.

1.11.2. 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.

1.12. ClearDERControl

1.12.1. ClearDERControlRequest

Class

Field Name	Field Type	Card.	Description
isDefault	boolean	1..1	Required. True: clearing default DER controls. False: clearing scheduled controls.
controlType	DERControlEnumType	0..1	Optional. Name of control settings to clear. Not used when *controlId* is provided.
controlId	identifierString[0..36]	0..1	Optional. Id of control setting to clear. When omitted all settings for *controlType* are cleared.

1.12.2. ClearDERControlResponse

Class

Field Name	Field Type	Card.	Description
status	DERControlStatusEnumType	1..1	Required. Result of operation.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information

1.13. ClearDisplayMessage

1.13.1. 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 O05 - Clear a Display Message.

Class

Field Name	Field Type	Card.	Description
id	integer, 0 ⇐ val	1..1	Required. Id of the message that SHALL be removed from the Charging Station.

1.13.2. 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 O05 - 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.

1.14. ClearedChargingLimit

1.14.1. 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	string[0..20]	1..1	Required. Source of the charging limit. Allowed values defined in Appendix as ChargingLimitSourceEnumStringType.
evseId	integer, 0 <= val	0..1	Optional. EVSE Identifier.

1.14.2. ClearedChargingLimitResponse

This contains the field definition of the ClearedChargingLimitResponse PDU sent by the CSMS to the Charging Station. No fields are defined.

1.15. ClearTariffs

1.15.1. ClearTariffsRequest

Class

Field Name	Field Type	Card.	Description
tariffIds	string[0..60]	0..*	Optional. List of tariff Ids to clear. When absent clears all tariffs at *evseId*.
evseId	integer, 0 <= val	0..1	Optional. When present only clear tariffs matching *tariffIds* at EVSE *evseId*.

1.15.2. ClearTariffsResponse

Class

Field Name	Field Type	Card.	Description
clearTariffsResult	ClearTariffsResultType	1..*	Required. Result per tariff.

1.16. ClearVariableMonitoring

1.16.1. 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, 0 <= val	1..*	Required. List of the monitors to be cleared, identified by there Id.

1.16.2. 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 status per monitor.

1.17. ClosePeriodicEventStream

1.17.1. ClosePeriodicEventStreamRequest

Class

Field Name	Field Type	Card.	Description
id	integer, 0 <= val	1..1	Required. Id of stream to close.

1.17.2. ClosePeriodicEventStreamResponse

1.18. CostUpdated

1.18.1. 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.

1.18.2. 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.

1.19. CustomerInformation

This contains the field definition of the CustomerInformationRequest PDU sent by the CSMS to the Charging Station.

1.19.1. CustomerInformationRequest

Class

Field Name	Field Type	Card.	Description
requestId	integer, 0 <= val	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.

Field Name	Field Type	Card.	Description
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.

1.19.2. 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.

1.20. DataTransfer

1.20.1. 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

1.20.2. 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.

1.21. DeleteCertificate

1.21.1. 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.

1.21.2. 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.

1.22. FirmwareStatusNotification

1.22.1. 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.
statusInfo	StatusInfoType	0..1	Optional. **(2.1)** Detailed status info

1.22.2. 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.

1.23. Get15118EVCertificate

1.23.1. 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..11000]	1..1	Required. **(2.1)** Raw CertificateInstallationReq request from EV, Base64 encoded. Extended to support ISO 15118-20 certificates. The minimum supported length is 11000. If a longer *exiRequest* is supported, then the supported length must be communicated in variable OCPPCommCtrlr.FieldLength["Get15118EVCertificateRequest.exiRequest"].
maximumContractCertificateChains	integer, 0 <= val	0..1	Optional. **(2.1)** Absent during ISO 15118-2 session. Required during ISO 15118-20 session. Maximum number of contracts that EV wants to install.
prioritizedEMAIDs	identifierString[0..255]	0..8	Optional. **(2.1)** Absent during ISO 15118-2 session. Optional during ISO 15118-20 session. List of EMAIDs for which contract certificates must be requested first, in case there are more certificates than allowed by *maximumContractCertificateChains*.

1.23.2. 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..17000]	1..1	Required. **(2/1)** Raw CertificateInstallationRes response for the EV, Base64 encoded. Extended to support ISO 15118-20 certificates. The minimum supported length is 17000. If a longer *exiResponse* is supported, then the supported length must be communicated in variable OCPPCommCtrlr.FieldLength["Get15118EVCertificateResponse.exiResponse"].
remainingContracts	integer, 0 <= val	0..1	Optional. **(2.1)** Number of contracts that can be retrieved with additional requests.
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.24. GetBaseReport

1.24.1. 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.

1.24.2. 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.

1.25. GetCertificateChainStatus

1.25.1. GetCertificateChainStatusRequest

Class

Field Name	Field Type	Card.	Description
certificateStatusRequests	CertificateStatusRequestInfoType	1..4	Required. Certificate to check revocation status for.

1.25.2. GetCertificateChainStatusResponse

Class

Field Name	Field Type	Card.	Description
certificateStatus	CertificateStatusType	1..4	Required. Status of the certificate revocation check.

1.26. GetCertificateStatus

1.26.1. 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.

1.26.2. 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..18000]	0..1	Optional. **(2.1)** 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. The minimum supported length is 18000. If a longer *ocspResult* is supported, then the supported length must be communicated in variable OCPPCommCtrlr.FieldLength["GetCertificateStatusResponse.ocspResult"].

Field Name	Field Type	Card.	Description
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.

1.27. GetChargingProfiles

1.27.1. 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.
evseId	integer, 0 <= val	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 *chargingProfile*.
chargingProfile	ChargingProfileCriterionType	1..1	Required. Specifies the charging profile.

1.27.2. 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.

1.28. GetCompositeSchedule

1.28.1. 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.
evseId	integer, 0 <= val	1..1	Required. The ID of the EVSE for which the schedule is requested. When evseid=0, the Charging Station will calculate the expected consumption for the grid connection.

1.28.2. 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
statusInfo	StatusInfoType	0..1	Optional. Detailed status information.
schedule	CompositeScheduleType	0..1	Optional. This field contains the calculated composite schedule. It may only be omitted when this message contains status Rejected.

1.29. GetDERControl

1.29.1. GetDERControlRequest

Class

Field Name	Field Type	Card.	Description
requestId	integer	1..1	Required. RequestId to be used in ReportDERControlRequest.
isDefault	boolean	0..1	Optional. True: get a default DER control. False: get a scheduled control.
controlType	DERControlEnumType	0..1	Optional. Type of control settings to retrieve. Not used when *controlId* is provided.
controlId	identifierString[0..36]	0..1	Optional. Id of setting to get. When omitted all settings for *controlType* are retrieved.

1.29.2. GetDERControlResponse

Class

Field Name	Field Type	Card.	Description
status	DERControlStatusEnumType	1..1	Required. Result of operation.
statusInfo	StatusInfoType	0..1	Optional. Detailed status info.

1.30. GetDisplayMessages

1.30.1. GetDisplayMessagesRequest

Class

Field Name	Field Type	Card.	Description
id	integer, 0 ⇐ val	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.

1.30.2. 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.

1.31. GetInstalledCertificateIds

1.31.1. 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.

1.31.2. 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.

1.32. GetLocalListVersion

1.32.1. GetLocalListVersionRequest

This contains the field definition of the GetLocalListVersionRequest PDU sent by the CSMS to the Charging Station. No fields are defined.

1.32.2. 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.

1.33. GetLog

1.33.1. 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 ⇐ val	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.

1.33.2. 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.

1.34. GetMonitoringReport

1.34.1. 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.

1.34.2. 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.

1.35. GetPeriodicEventStream

1.35.1. GetPeriodicEventStreamRequest

1.35.2. GetPeriodicEventStreamResponse

Class

Field Name	Field Type	Card.	Description
constantStreamData	ConstantStreamDataType	0..*	Optional. List of constant part of streams

1.36. GetReport

1.36.1. 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.

1.36.2. 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.

1.37. GetTariffs

1.37.1. GetTariffsRequest

Class

Field Name	Field Type	Card.	Description
evseId	integer, 0 <= val	1..1	Required. EVSE id to get tariff from. When *evseId* = 0, this gets tariffs from all EVSEs.

1.37.2. GetTariffsResponse

Class

Field Name	Field Type	Card.	Description
status	TariffGetStatusEnumType	1..1	Required. Status of operation

Field Name	Field Type	Card.	Description
tariffAssignments	TariffAssignmentType	0..*	Optional. Installed default and user-specific tariffs per EVSE
statusInfo	StatusInfoType	0..1	Optional. Details status information

1.38. GetTransactionStatus

1.38.1. 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.

1.38.2. 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.

1.39. GetVariables

1.39.1. 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.

1.39.2. 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.

1.40. Heartbeat

1.40.1. HeartbeatRequest

This contains the field definition of the HeartbeatRequest PDU sent by the Charging Station to the CSMS. No fields are defined.

1.40.2. 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.

1.41. InstallCertificate

1.41.1. 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..10000]	1..1	Required. A PEM encoded X.509 certificate.

1.41.2. 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.

1.42. LogStatusNotification

1.42.1. 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.
statusInfo	StatusInfoType	0..1	Optional. **(2.1)** Detailed status info

1.42.2. 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.

1.43. MeterValues

1.43.1. 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
evseId	integer, 0 ⇐ val	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. The following Configuration Variables are used to configure which measurands are sent: - `AlignedDataMeasurands` - `AlignedDataUpstreamMeasurands`

1.43.2. 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.

1.44. NotifyAllowedEnergyTransfer

1.44.1. NotifyAllowedEnergyTransferRequest

(2.1) Message sent by CSMS to update the list of authorized energy services, e.g. to allow bidirectional charging for a charging session that is already in progress. One example is that the EV has already started a transaction in charging-only mode and meanwhile the CSMS has found that this EV is authorized by some secondary actor, such as an aggregating party, to use bidirectional charging. This message is then used to give the EV the opportunity to change energy service from charging-only to bidirectional charging.

Another example is that the CSMS wishes to change the active energy service. This is done by updating the list of authorized energy services and omitting the currently active energy service.

Class

Field Name	Field Type	Card.	Description
transactionId	identifierString[0..36]	1..1	Required. The transaction for which the allowed energy transfer is allowed.
allowedEnergyTransfer	EnergyTransferModeEnumType	1..*	Required. Modes of energy transfer that are accepted by CSMS.

1.44.2. NotifyAllowedEnergyTransferResponse

(2.1) Status of NotifyAllowedEnergyServicesRequest. Request should normally not be rejected, unless there are some technical problems.

Class

Field Name	Field Type	Card.	Description
status	NotifyAllowedEnergyTransferStatusEnumType	1..1	Required.
statusInfo	StatusInfoType	0..1	Optional.

1.45. NotifyChargingLimit

1.45.1. 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
evseId	integer, 0 <= val	0..1	Optional. The EVSE to which the charging limit is set. If absent or when zero, it applies to the entire Charging Station.
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.

1.45.2. NotifyChargingLimitResponse

The NotifyChargingLimitResponse message is sent by the CSMS to the Charging Station in response to a NotifyChargingLimitsRequest. No fields are defined.

1.46. NotifyCustomerInformation

This contains the field definition of the NotifyCustomerInformationRequest PDU sent by the Charging Station to the CSMS.

1.46.1. 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, 0 <= val	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, 0 <= val	1..1	Required. The Id of the request.

1.46.2. NotifyCustomerInformationResponse

1.47. NotifyDERAlarm

1.47.1. NotifyDERAlarmRequest

Class

Field Name	Field Type	Card.	Description
controlType	DERControlEnumType	1..1	Required. Name of DER control, e.g. LFMustTrip
gridEventFault	GridEventFaultEnumType	0..1	Optional. Type of grid event that caused this

Field Name	Field Type	Card.	Description
alarmEnded	boolean	0..1	Optional. True when error condition has ended. Absent or false when alarm has started.
timestamp	dateTime	1..1	Required. Time of start or end of alarm.
extraInfo	string[0..200]	0..1	Optional. Optional info provided by EV.

1.47.2. NotifyDERAlarmResponse

This message has no fields.

1.48. NotifyDERStartStop

1.48.1. NotifyDERStartStopRequest

Class

Field Name	Field Type	Card.	Description
controlId	identifierString[0..36]	1..1	Required. Id of the started or stopped DER control. Corresponds to the *controlId* of the SetDERControlRequest.
started	boolean	1..1	Required. True if DER control has started. False if it has ended.
timestamp	dateTime	1..1	Required. Time of start or end of event.
supersededIds	identifierString[0..36]	0..24	Optional. List of controlIds that are superseded as a result of this control starting.

1.48.2. NotifyDERStartStopResponse

This message has no fields.

1.49. NotifyDisplayMessages

1.49.1. 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.

1.49.2. NotifyDisplayMessagesResponse

The NotifyDisplayMessagesResponse message is sent by the CSMS to the Charging Station in response to a NotifyDisplayMessagesRequest. No fields are defined.

1.50. NotifyEVChargingNeeds

1.50.1. 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
evseId	integer, 1 <= val	1..1	Required. Defines the EVSE and connector to which the EV is connected. EvseId may not be 0.
maxScheduleTuples	integer, 0 <= val	0..1	Optional. Contains the maximum elements the EV supports for: - ISO 15118-2: schedule tuples in SASchedule (both Pmax and Tariff). - ISO 15118-20: PowerScheduleEntry, PriceRule and PriceLevelScheduleEntries.
timestamp	dateTime	0..1	Optional. **(2.1)** Time when EV charging needs were received. Field can be added when charging station was offline when charging needs were received.
chargingNeeds	ChargingNeedsType	1..1	Required. The characteristics of the energy delivery required.

1.50.2. 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.

1.51. NotifyEVChargingSchedule

1.51.1. 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.
evseId	integer, 1 <= val	1..1	Required. The charging schedule contained in this notification applies to an EVSE. EvseId must be > 0.
selectedChargingScheduleId	integer, 0 <= val	0..1	Optional. **(2.1)** Id of the *chargingSchedule* that EV selected from the provided ChargingProfile.
powerToleranceAcceptance	boolean	0..1	Optional. **(2.1)** True when power tolerance is accepted by EV. This value is taken from EVPowerProfile.PowerToleranceAcceptance in the ISO 15118-20 PowerDeliverReq message..
chargingSchedule	ChargingScheduleType	1..1	Required. Planned energy consumption of the EV over time. Always relative to timeBase.

1.51.2. 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.

1.52. NotifyEvent

1.52.1. 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, 0 ⇐ val	1..1	Required. Sequence number of this message. First message starts at 0.
eventData	EventDataType	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 usally 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.

1.52.2. NotifyEventResponse

Response to NotifyEventRequest. No fields are defined.

1.53. NotifyMonitoringReport

1.53.1. 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, 0 ⇐ val	1..1	Required. Sequence number of this message. First message starts at 0.

Field Name	Field Type	Card.	Description
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.

1.53.2. NotifyMonitoringReportResponse

Response to a NotifyMonitoringRequest message. No fields are defined.

1.54. NotifyPeriodicEventStream

This is a message of *messageType* SEND. It does not have a response.

Class

Field Name	Field Type	Card.	Description
id	integer, 0 <= val	1..1	Required. Id of stream.
pending	integer, 0 <= val	1..1	Required. Number of data elements still pending to be sent.
basetime	dateTime	1..1	Required. Base timestamp to add to time offset of values.
data	StreamDataElementType	1..*	Required. Variable part of stream data

1.55. NotifyPriorityCharging

1.55.1. NotifyPriorityChargingRequest

(2.1) Message sent by Charging Station to notify CSMS that it has switched to the priority charging profile, that allows for the maximum possible current or power under current circumstances. Message contains a *transactionId*, because it only applies to the transaction in progress.

Class

Field Name	Field Type	Card.	Description
transactionId	identifierString[0..36]	1..1	Required. The transaction for which priority charging is requested.
activated	boolean	1..1	Required. True if priority charging was activated. False if it has stopped using the priority charging profile.

1.55.2. NotifyPriorityChargingResponse

(2.1) This response message has an empty body.

1.56. NotifyReport

1.56.1. 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

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 notifyReportRequest message. Default value when omitted is false.
seqNo	integer, 0 ⇐ val	1..1	Required. Sequence number of this message. First message starts at 0.
reportData	ReportDataType	0..*	Optional. List of ReportData.

1.56.2. NotifyReportResponse

Response to a NotifyReportRequest message. No fields are defined.

1.57. NotifySettlement

1.57.1. NotifySettlementRequest

Class

Field Name	Field Type	Card.	Description
transactionId	identifierString[0..36]	0..1	Optional. The *transactionId* that the settlement belongs to. Can be empty if the payment transaction is canceled prior to the start of the OCPP transaction.
pspRef	identifierString[0..255]	1..1	Required. The payment reference received from the payment terminal and is used as the value for *idToken*.
status	PaymentStatusEnumType	1..1	Required. The status of the settlement attempt.
statusInfo	string[0..500]	0..1	Optional. Additional information from payment terminal/payment process.
settlementAmount	decimal	1..1	Required. The amount that was settled, or attempted to be settled (in case of failure).
settlementTime	dateTime	1..1	Required. The time when the settlement was done.
receiptId	string[0..50]	0..1	Optional.
receiptUrl	string[0..2000]	0..1	Optional. The receipt URL, to be used if the receipt is generated by the payment terminal or the CS.
vatNumber	string[0..20]	0..1	Optional. VAT number for a company receipt.
vatCompany	AddressType	0..1	Optional. Company address associated with VAT number.

1.57.2. NotifySettlementResponse

Class

Field Name	Field Type	Card.	Description
receiptUrl	string[0..2000]	0..1	Optional. The receipt URL if receipt generated by CSMS. The Charging Station can QR encode it and show it to the EV Driver.
receiptId	string[0..50]	0..1	Optional. The receipt id if the receipt is generated by CSMS.

1.58. NotifyWebPaymentStarted

1.58.1. NotifyWebPaymentStartedRequest

Class

Field Name	Field Type	Card.	Description
evseId	integer, 0 ⇐ val	1..1	Required. EVSE id for which transaction is requested.
timeout	integer	1..1	Required. Timeout value in seconds after which no result of web payment process (e.g. QR code scanning) is to be expected anymore.

1.58.2. NotifyWebPaymentStartedResponse

1.59. OpenPeriodicEventStream

1.59.1. OpenPeriodicEventStreamRequest

Class

Field Name	Field Type	Card.	Description
constantStreamData	ConstantStreamDataType	1..1	Required. Constant part of stream data

1.59.2. OpenPeriodicEventStreamResponse

Class

Field Name	Field Type	Card.	Description
status	GenericStatusEnumType	1..1	Required. Result of request.
statusInfo	StatusInfoType	0..1	Optional. Detailed status info

1.60. PublishFirmware

1.60.1. 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..2000]	1..1	Required. This contains a string containing a URI pointing to a location from which to retrieve the firmware.
retries	integer, 0 ⇐ val	0..1	Optional. This specifies how many times 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, 0 ⇐ val	1..1	Required. The Id of the request.
retryInterval	integer, 0 ⇐ val	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.

1.60.2. 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.

1.61. PublishFirmwareStatusNotification

1.61.1. 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..2000]	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 <= val	0..1	Optional. The request id that was provided in the PublishFirmwareRequest which triggered this action.
statusInfo	StatusInfoType	0..1	Optional. **(2.1)** Detailed status info

1.61.2. PublishFirmwareStatusNotificationResponse

This contains the field definition of the PublishFirmwareStatusNotificationResponse PDU sent by the CSMS to the Charging station in response to a PublishFirmwareStatusNotificationRequest.

1.62. PullDynamicScheduleUpdate

1.62.1. PullDynamicScheduleUpdateRequest

(2.1) This message is sent by a Charging Station to request an update of setpoints and/or limits of the charging profile with given *chargingProfileId*.

Class

Field Name	Field Type	Card.	Description
chargingProfileId	integer	1..1	Required. Id of charging profile to update.

1.62.2. PullDynamicScheduleUpdateResponse

(2.1) If no data can be provided by CSMS, then the response will only contain *status*.

Class

Field Name	Field Type	Card.	Description
status	ChargingProfileStatusEnumType	1..1	Required. Result of request.
scheduleUpdate	ChargingScheduleUpdateType	0..1	Optional. Updated charging schedule period values.
statusInfo	StatusInfoType	0..1	Optional. Additional info about status

1.63. ReportChargingProfiles

1.63.1. 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 to 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	string[0..20]	1..1	Required. Source that has installed this charging profile. Values defined in Appendix as ChargingLimitSourceEnumStringType.
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.
evseId	integer, 0 ⇐ val	1..1	Required. The evse to which the charging profile applies. If evseId = 0, the message contains an overall limit for the Charging Station.
chargingProfile	ChargingProfileType	1..*	Required. The charging profile as configured in the Charging Station.

1.63.2. ReportChargingProfilesResponse

The ReportChargingProfilesResponse message is sent by the CSMS to the Charging Station in response to a ReportChargingProfilesRequest. No fields are defined.

1.64. ReportDERControl

1.64.1. ReportDERControlRequest

Reports DER controls requested by a GetDERControlRequest message. The report may consist of more than one message.

Class

Field Name	Field Type	Card.	Description
requestId	integer	1..1	Required. RequestId from GetDERControlRequest.
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.
fixedPFAbsorb	FixedPFGetType	0..24	Optional. Fixed power factor setpoint when absorbing active power
fixedPFInject	FixedPFGetType	0..24	Optional. Fixed power factor setpoint when injecting active power
fixedVar	FixedVarGetType	0..24	Optional. Fixed reactive power setting
limitMaxDischarge	LimitMaxDischargeGetType	0..24	Optional. Limit maximum discharge as percentage of rated capability
freqDroop	FreqDroopGetType	0..24	Optional. Frequency-Watt parameterized mode
enterService	EnterServiceGetType	0..24	Optional. Enter service after trip parameters
gradient	GradientGetType	0..24	Optional. Gradient settings

Field Name	Field Type	Card.	Description
curve	DERCurveGetType	0..24	Optional. Voltage/Frequency/Active/Reactive curve

1.64.2. ReportDERControlResponse

This is an empty message sent by CSMS in response to a ReportDERControlRequest message.

1.65. RequestBatterySwap

1.65.1. RequestBatterySwapRequest

Class

Field Name	Field Type	Card.	Description
requestId	integer	1..1	Required. Request id to match with BatterySwapRequest.
idToken	IdTokenType	1..1	Required. Id token of EV driver.

1.65.2. RequestBatterySwapResponse

Class

Field Name	Field Type	Card.	Description
status	GenericStatusEnumType	1..1	Required. Accepted or rejected the request.
statusInfo	StatusInfoType	0..1	Optional. Additional info on status

1.66. RequestStartTransaction

1.66.1. RequestStartTransactionRequest

This contains the field definitions of the RequestStartTransactionRequest PDU sent to Charging Station by CSMS.

Class

Field Name	Field Type	Card.	Description
evseId	integer, 1 ⇐ val	0..1	Optional. Number of the EVSE on which to start the transaction. EvseId 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. Use to start a transaction.
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).

1.66.2. 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.

1.67. RequestStopTransaction

1.67.1. 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.

1.67.2. 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.

1.68. ReservationStatusUpdate

1.68.1. 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, 0 ⇐ val	1..1	Required. The ID of the reservation.
reservationUpdateStatus	ReservationUpdateStatusEnumType	1..1	Required. The updated reservation status.

1.68.2. 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.

1.69. ReserveNow

1.69.1. 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, 0 ⇐ val	1..1	Required. Id of reservation.
expiryDateTime	dateTime	1..1	Required. Date and time at which the reservation expires.
connectorType	string[0..20]	0..1	Optional. This field specifies the connector type. Values defined in Appendix as ConnectorEnumStringType.
evseId	integer, 0 ⇐ val	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.

1.69.2. 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.

1.70. Reset

1.70.1. 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.
evseId	integer, 0 ⇐ val	0..1	Optional. This contains the ID of a specific EVSE that needs to be reset, instead of the entire Charging Station.

1.70.2. 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.

1.71. SecurityEventNotification

1.71.1. 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.

1.71.2. SecurityEventNotificationResponse

Sent by the CSMS to the Charging Station to confirm the receipt of a SecurityEventNotificationRequest message. No fields are defined.

1.72. SendLocalList

1.72.1. 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.

1.72.2. 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.

1.73. SetChargingProfile

1.73.1. 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
evseId	integer, 0 <= val	1..1	Required. For TxDefaultProfile an evseId=0 applies the profile to each individual evse. For ChargingStationMaxProfile and ChargingStationExternalConstraints an evseId=0 contains an overal limit for the whole Charging Station.
chargingProfile	ChargingProfileType	1..1	Required. The charging profile to be set at the Charging Station.

1.73.2. 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.

1.74. SetDefaultTariff

1.74.1. SetDefaultTariffRequest

Class

Field Name	Field Type	Card.	Description
evseId	integer, 0 <= val	1..1	Required. EVSE that tariff applies to. When *evseId* = 0, then tarriff applies to all EVSEs.
tariff	TariffType	1..1	Required. Tariff structure.

1.74.2. SetDefaultTariffResponse

Class

Field Name	Field Type	Card.	Description
status	TariffSetStatusEnumType	1..1	Required.
statusInfo	StatusInfoType	0..1	Optional. Detailed info on status

1.75. SetDERControl

1.75.1. SetDERControlRequest

Class

Field Name	Field Type	Card.	Description
isDefault	boolean	1..1	Required. True if this is a default DER control
controlId	identifierString[0..36]	1..1	Required. Unique id of this control, e.g. UUID
controlType	DERControlEnumType	1..1	Required. Type of control. Determines which setting field below is used.
curve	DERCurveType	0..1	Optional. Voltage/Frequency/Active/Reactive curve
fixedPFAbsorb	FixedPFType	0..1	Optional. Fixed power factor setpoint when absorbing active power
fixedPFInject	FixedPFType	0..1	Optional. Fixed power factor setpoint when injecting active power
fixedVar	FixedVarType	0..1	Optional. Fixed reactive power
limitMaxDischarge	LimitMaxDischargeType	0..1	Optional. Limit maximum discharge as percentage of rated capability
freqDroop	FreqDroopType	0..1	Optional. Frequency-Watt parameterized mode
enterService	EnterServiceType	0..1	Optional. Enter service after trip parameters (default control only)
gradient	GradientType	0..1	Optional. Gradient (default ramp rate) settings (default control only)

1.75.2. SetDERControlResponse

Class

Field Name	Field Type	Card.	Description
status	DERControlStatusEnumType	1..1	Required. Result of operation.
supersededIds	identifierString[0..36]	0..24	Optional. List of controlIds that are superseded as a result of setting this control.
statusInfo	StatusInfoType	0..1	Optional. Additional details on status

1.76. SetDisplayMessage

1.76.1. 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 O01 - Set Display Message, O02 - Set Display Message for Transaction and O06 - 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.

1.76.2. 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 O01 - Set Display Message, O02 - Set Display Message for Transaction and O06 - 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.

1.77. SetMonitoringBase

1.77.1. 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

1.77.2. 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.

1.78. SetMonitoringLevel

1.78.1. 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, 0 ⇐ val	1..1	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. The severity levels have the following meaning: **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.

1.78.2. 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.

1.79. SetNetworkProfile

1.79.1. 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.

1.79.2. 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.

1.80. SetVariableMonitoring

1.80.1. 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.

1.80.2. 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.

1.81. SetVariables

1.81.1. 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.

1.81.2. 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.

1.82. SignCertificate

1.82.1. 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.
requestId	integer	0..1	Optional. **(2.1)** RequestId to match this message with the CertificateSignedRequest.
hashRootCertificate	CertificateHashDataType	0..1	Optional. **(2.1)** The hash of the root certificate to identify the PKI to use.

1.82.2. 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.

1.83. StatusNotification

1.83.1. 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.
evseId	integer, 0 <= val	1..1	Required. The id of the EVSE to which the connector belongs for which the the status is reported.
connectorId	integer, 0 <= val	1..1	Required. The id of the connector within the EVSE for which the status is reported.

1.83.2. 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.

1.84. TransactionEvent

1.84.1. 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, 0 ⇐ val	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 ⇐ val >= 3	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 as follows: 1: The numberPhases in the currently used ChargingSchedule. 2: The number of phases provided via device management.
cableMaxCurrent	integer	0..1	Optional. The maximum current of the connected cable in Ampere (A).
reservationId	integer, 0 ⇐ val	0..1	Optional. This contains the Id of the reservation that terminates as a result of this transaction.
preconditioningStatus	PreconditioningStatusEnumType	0..1	Optional. **(2.1)** The current preconditioning status of the BMS in the EV. Default value is Unknown.
evseSleep	boolean	0..1	Optional. **(2.1)** True when EVSE electronics are in sleep mode for this transaction. Default value (when absent) is false.
meterValue	MeterValueType	0..*	Optional. This contains the meter values relevant to the transaction. Depending on the *eventType* of this TransactionEventRequest the following Configuration Variable is used to configure which measurands are used: Started: `SampledDataTxStartedMeasurands` Updated: `SampledDataTxUpdatedMeasurands` and `AlignedDataMeasurands` Ended: `SampledDataTxEndedMeasurands` and `AlignedDataTxEndedMeasurands`

Field Name	Field Type	Card.	Description
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 *idTokenInfo* in the TransactionEventResponse. *idToken* should not be present in the TransactionEventRequest when a transaction is ended by a RequestStopTransactionRequest or a ResetRequest.
evse	EVSEType	0..1	Optional. This identifies which evse (and connector) of the Charging Station is used.
transactionInfo	TransactionType	1..1	Required. Contains transaction specific information.
costDetails	CostDetailsType	0..1	Optional. **(2.1)** Optional. Only required in TransactionEventRequest('Ended') and only if Charging Station calculated cost locally.

1.84.2. 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 running cost. When *eventType* of TransactionEventRequest is Ended, then this contains the final total cost 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 has a higher priority than 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 .
updatedPersonalMessageExtra	MessageContentType	0..4	Optional. **(2.1)** Additional languages besides the default language in *updatedPersonalMessage*.
transactionLimit	TransactionLimitType	0..1	Optional. **(2.1)** Maximum cost/energy/time limit allowed for this transaction.

1.85. TriggerMessage

1.85.1. 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.
customTrigger	string[0..50]	0..1	Optional. **(2.1)** When *requestedMessage* = `CustomTrigger` this will trigger sending the corresponding message in field *customTrigger*, if supported by Charging Station.
evse	EVSEType	0..1	Optional. Can be used to specifiy the EVSE and Connector if required for the message which needs to be sent.

1.85.2. 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.

1.86. UnlockConnector

1.86.1. 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
evseId	integer, 0 <= val	1..1	Required. This contains the identifier of the EVSE for which a connector needs to be unlocked.
connectorId	integer, 0 <= val	1..1	Required. This contains the identifier of the connector that needs to be unlocked.

1.86.2. 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.

1.87. UnpublishFirmware

1.87.1. 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.

1.87.2. 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.

1.88. UpdateDynamicSchedule

1.88.1. UpdateDynamicScheduleRequest

(2.1) This message is used to update a setpoint or limit in a dynamic charging profile.

Class

Id of dynamic charging profile to update.

Field Name	Field Type	Card.	Description
chargingProfileId	integer	1..1	Required. Id of charging profile to update.
scheduleUpdate	ChargingScheduleUpdateType	1..1	Required. Updated values for charging schedule period.

1.88.2. UpdateDynamicScheduleResponse

(2.1) Returns whether the Charging Station has been able to process the message successfully.

Class

Field Name	Field Type	Card.	Description
status	ChargingProfileStatusEnumType	1..1	Required. Returns whether message was processed successfully.
statusInfo	StatusInfoType	0..1	Optional. Detailed status info.

1.89. UpdateFirmware

1.89.1. 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 ⇐ val	0..1	Optional. This specifies how many times 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.

Field Name	Field Type	Card.	Description
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.

1.89.2. 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.

1.90. UsePriorityCharging

1.90.1. UsePriorityChargingRequest

(2.1) Message sent by CSMS to tell Charging Station to switch to the priority charging profile, that allows for the maximum possible current or power under current circumstances. Message contains a *transactionId*, because it only applies to the transaction in progress.

Class

Field Name	Field Type	Card.	Description
transactionId	identifierString[0..36]	1..1	Required. The transaction for which priority charging is requested.
activate	boolean	1..1	Required. True to request priority charging. False to request stopping priority charging.

1.90.2. UsePriorityChargingResponse

(2.1) Status of the UsePriorityChargingRequest.

Class

Field Name	Field Type	Card.	Description
status	PriorityChargingStatusEnumType	1..1	Required. Result of the request.
statusInfo	StatusInfoType	0..1	Optional.

1.91. VatNumberValidation

1.91.1. VatNumberValidationRequest

Class

Field Name	Field Type	Card.	Description
vatNumber	string[0..20]	1..1	Required. VAT number to check.
evseId	integer, 0 ⇐ val	0..1	Optional. EVSE id for which check is done

1.91.2. VatNumberValidationResponse

Class

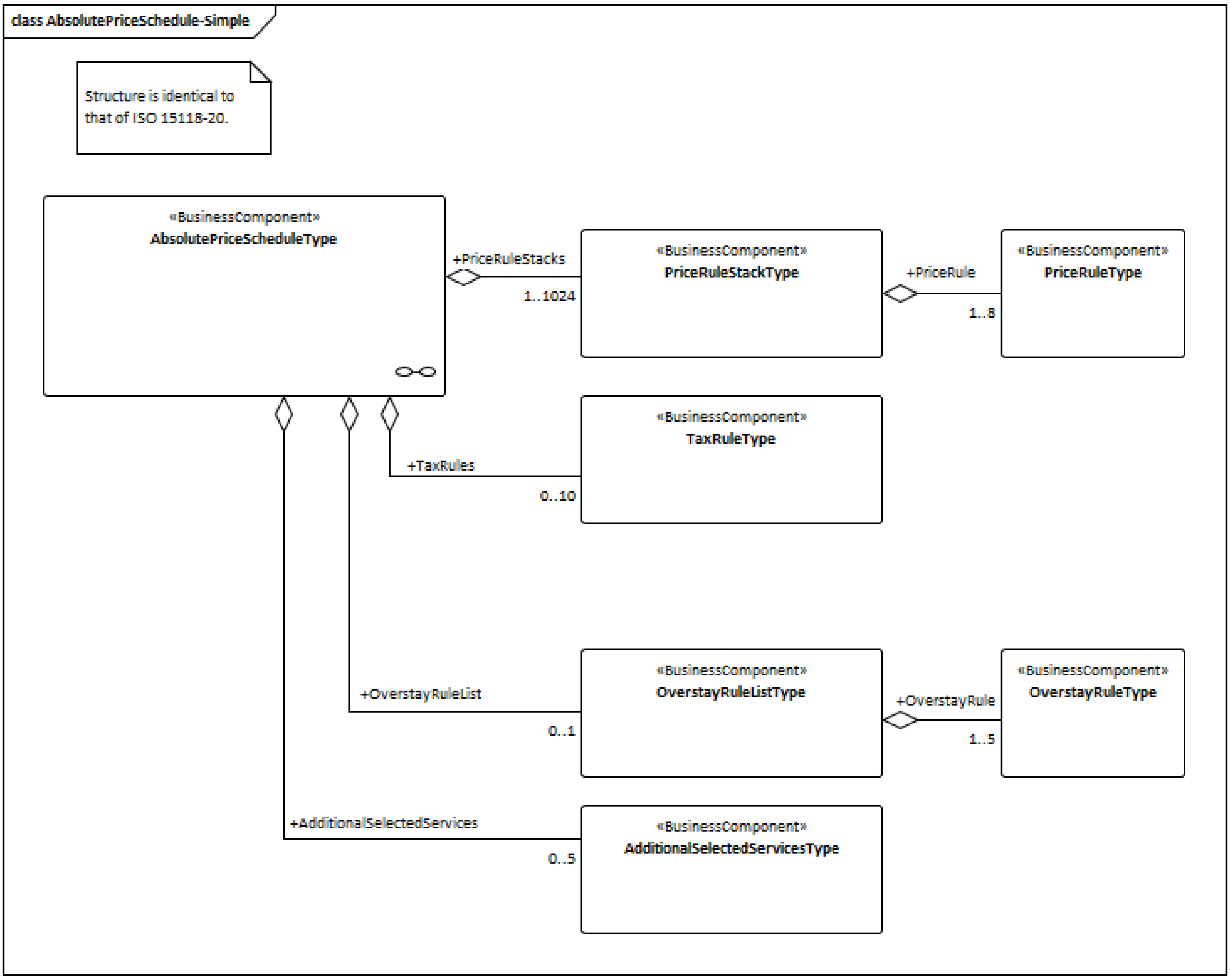
Field Name	Field Type	Card.	Description
vatNumber	string[0..20]	1..1	Required. VAT number that was requested.
evseId	integer, 0 <= val	0..1	Optional. EVSE id for which check was requested.
status	GenericStatusEnumType	1..1	Required. Result of operation.
company	AddressType	0..1	Optional. Company address associated with *vatNumber*.
statusInfo	StatusInfoType	0..1	Optional. Additional info on status

Chapter 2. Datatypes

2.1. AbsolutePriceScheduleType

Class

The AbsolutePriceScheduleType is modeled after the same type that is defined in ISO 15118-20, such that if it is supplied by an EMSP as a signed EXI message, the conversion from EXI to JSON (in OCPP) and back to EXI (for ISO 15118-20) does not change the digest and therefore does not invalidate the signature.


class AbsolutePriceSchedule-Simple
Structure is identical to that of ISO 15118-20.
«BusinessComponent» AbsolutePriceScheduleType
+PriceRuleStacks
1..1024
«BusinessComponent» PriceRuleStackType
+PriceRule
1..8
«BusinessComponent» PriceRuleType
+TaxRules
0..10
«BusinessComponent» TaxRuleType
+OverstayRuleList
0..1
«BusinessComponent» OverstayRuleListType
+OverstayRule
1..5
«BusinessComponent» OverstayRuleType
+AdditionalSelectedServices
0..5
«BusinessComponent» AdditionalSelectedServicesType

AbsolutePriceScheduleType is used by: Common:ChargingScheduleType

Field Name	Field Type	Card.	Description
timeAnchor	dateTime	1..1	Required. Starting point of price schedule.
priceScheduleID	integer, 0 < = val	1..1	Required. Unique ID of price schedule
priceScheduleDescription	string[0..160]	0..1	Optional. Description of the price schedule.
currency	string[0..3]	1..1	Required. Currency according to ISO 4217.
language	string[0..8]	1..1	Required. String that indicates what language is used for the human readable strings in the price schedule. Based on ISO 639.
priceAlgorithm	string[0..2000]	1..1	Required. A string in URN notation which shall uniquely identify an algorithm that defines how to compute an energy fee sum for a specific power profile based on the EnergyFee information from the PriceRule elements.

Field Name	Field Type	Card.	Description
priceRuleStacks	PriceRuleStackType	1..1024	Required. A set of pricing rules for parking and energy costs.
taxRules	TaxRuleType	0..10	Optional. Optional. Describes the applicable tax rule(s) for this price schedule.
additionalSelectedServices	AdditionalSelectedServicesType	0..5	Optional. Optional. A set of prices for optional services (e.g. valet, carwash).
overstayRuleList	OverstayRuleListType	0..1	Optional. Optional. A set of overstay rules that allows for escalation of charges after the overstay is triggered.
minimumCost	RationalNumberType	0..1	Optional. Optional. Minimum amount to be billed for the overall charging session (e.g. including energy, parking, and overstay).
maximumCost	RationalNumberType	0..1	Optional. Optional. Maximum amount to be billed for the overall charging session (e.g. including energy, parking, and overstay).

2.2. ACChargingParametersType

Class

EV AC charging parameters for ISO 15118-2

ACChargingParametersType is used by: Common:ChargingNeedsType

Field Name	Field Type	Card.	Description
energyAmount	decimal	1..1	Required. Amount of energy requested (in Wh). This includes energy required for preconditioning. Relates to: **ISO 15118-2**: AC_EVChargeParameterType: EAmount **ISO 15118-20**: Dynamic/Scheduled_SEReqControlModeType: EVTargetEnergyRequest
evMinCurrent	decimal	1..1	Required. Minimum current (amps) supported by the electric vehicle (per phase). Relates to: **ISO 15118-2**: AC_EVChargeParameterType: EVMinCurrent
evMaxCurrent	decimal	1..1	Required. Maximum current (amps) supported by the electric vehicle (per phase). Includes cable capacity. Relates to: **ISO 15118-2**: AC_EVChargeParameterType: EVMaxCurrent
evMaxVoltage	decimal	1..1	Required. Maximum voltage supported by the electric vehicle. Relates to: **ISO 15118-2**: AC_EVChargeParameterType: EVMaxVoltage

2.3. 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..255]	1..1	Required. **(2.1)** This field specifies the additional IdToken.

Field Name	Field Type	Card.	Description
type	string[0..50]	1..1	Required. *additionalInfo* can be used to send extra information to 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 the *type* is not supported, the CSMS/Charging Station MAY ignore the *additionalInfo*.

2.4. AdditionalSelectedServicesType

Class

Part of ISO 15118-20 price schedule.

AdditionalSelectedServicesType is used by: Common:AbsolutePriceScheduleType

Field Name	Field Type	Card.	Description
serviceName	string[0..80]	1..1	Required. Human readable string to identify this service.
serviceFee	RationalNumberType	1..1	Required. Cost of the service.

2.5. AddressType

Class

(2.1) A generic address format.

AddressType is used by: NotifySettlementRequest , VatNumberValidationResponse

Field Name	Field Type	Card.	Description
name	string[0..50]	1..1	Required. Name of person/company
address1	string[0..100]	1..1	Required. Address line 1
address2	string[0..100]	0..1	Optional. Address line 2
city	string[0..100]	1..1	Required. City
postalCode	string[0..20]	0..1	Optional. Postal code
country	string[0..50]	1..1	Required. Country name

2.6. 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..2000]	1..1	Required. The Access Point Name as an URL.
apnUserName	string[0..50]	0..1	Optional. APN username.
apnPassword	string[0..64]	0..1	Optional. **(2.1)** APN Password.
simPin	integer	0..1	Optional. SIM card pin code.

Field Name	Field Type	Card.	Description
preferredNetwork	identifierString[0..6]	0..1	Optional. Preferred network, written as MCC and MNC concatenated. See note.
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.

2.7. 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.

2.8. BatteryDataType

Class

BatteryDataType is used by: BatterySwapRequest

Field Name	Field Type	Card.	Description
evseId	integer, 0 < = val	1..1	Required. Slot number where battery is inserted or removed.
serialNumber	string[0..50]	1..1	Required. Serial number of battery.
soC	decimal, 0 < = val < = 100	1..1	Required. State of charge
soH	decimal, 0 < = val < = 100	1..1	Required. State of health
productionDate	dateTime	0..1	Optional. Production date of battery.
vendorInfo	string[0..500]	0..1	Optional. Vendor-specific info from battery in undefined format.

2.9. 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).

2.10. CertificateHashDataType

Class

CertificateHashDataType is used by: Common:CertificateHashDataChainType , Common:CertificateStatusRequestInfoType , Common:CertificateStatusType , SignCertificateRequest , 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.

2.11. CertificateStatusRequestInfoType

Class

Data necessary to request the revocation status of a certificate.

CertificateStatusRequestInfoType is used by: GetCertificateChainStatusRequest

Field Name	Field Type	Card.	Description
source	CertificateStatusSourceEnumType	1..1	Required. Source of status: OCSP, CRL
urls	string[0..2000]	1..5	Required. URL(s) of *source*.
certificateHashData	CertificateHashDataType	1..1	Required. Hash data of certificate.

2.12. CertificateStatusType

Class

Revocation status of certificate

CertificateStatusType is used by: GetCertificateChainStatusResponse

Field Name	Field Type	Card.	Description
source	CertificateStatusSourceEnumType	1..1	Required. Source of status: OCSP, CRL
status	CertificateStatusEnumType	1..1	Required. Status of certificate: good, revoked or unknown.
nextUpdate	dateTime	1..1	Required.
certificateHashData	CertificateHashDataType	1..1	Required. Hash data of the certificate.

2.13. ChargingLimitType

Class

ChargingLimitType is used by: NotifyChargingLimitRequest

Field Name	Field Type	Card.	Description
chargingLimitSource	string[0..20]	1..1	Required. Represents the source of the charging limit. Values defined in appendix as ChargingLimitSourceEnumStringType.

Field Name	Field Type	Card.	Description
isLocalGeneration	boolean	0..1	Optional. **(2.1)** True when the reported limit concerns local generation that is providing extra capacity, instead of a limitation.
isGridCritical	boolean	0..1	Optional. Indicates whether the charging limit is critical for the grid.

2.14. 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.
availableEnergyTransfer	EnergyTransferModeEnumType	0..*	Optional. **(2.1)** Modes of energy transfer that are marked as available by EV.
controlMode	ControlModeEnumType	0..1	Optional. **(2.1)** Indicates whether EV wants to operate in Dynamic or Scheduled mode. When absent, Scheduled mode is assumed for backwards compatibility. **ISO 15118-20:** ServiceSelectionReq(SelectedEnergyTransferService)
mobilityNeedsMode	MobilityNeedsModeEnumType	0..1	Optional. **(2.1)** Value of EVCC indicates that EV determines min/target SOC and departure time. A value of EVCC_SECC indicates that charging station or CSMS may also update min/target SOC and departure time. **ISO 15118-20:** ServiceSelectionReq(SelectedEnergyTransferService)
departureTime	dateTime	0..1	Optional. Estimated departure time of the EV. **ISO 15118-2:** AC/DC_EVChargeParameterType: DepartureTime **ISO 15118-20:** Dynamic/Scheduled_SEReqControlModeType: DepartureTIme
v2xChargingParameters	V2XChargingParametersType	0..1	Optional. **(2.1)** The list of charging parameters that apply to an ISO 15118-20 session or any other session that supports bidirectional charging.
dcChargingParameters	DCChargingParametersType	0..1	Optional. EV DC charging parameters
acChargingParameters	ACChargingParametersType	0..1	Optional. EV AC charging parameters.
evEnergyOffer	EVEnergyOfferType	0..1	Optional. **(2.1)** Discharging and associated price offered by EV. Schedule periods during which EV is willing to discharge have a negative value for power.
derChargingParameters	DERChargingParametersType	0..1	Optional. **(2.1)** Additional charging parameters for ISO 15118-20 AC bidirectional sessions with DER control (AC_BPT_DER)

2.15. ChargingPeriodType

Class

A ChargingPeriodType consists of a start time, and a list of possible values that influence this period, for example: amount of energy charged this period, maximum current during this period etc.

ChargingPeriodType is used by: Common:CostDetailsType

Field Name	Field Type	Card.	Description
tariffId	string[0..60]	0..1	Optional. Unique identifier of the Tariff that was used to calculate cost. If not provided, then cost was calculated by some other means.
startPeriod	dateTime	1..1	Required. Start timestamp of charging period. A period ends when the next period starts. The last period ends when the session ends.
dimensions	CostDimensionType	0..*	Optional. List of volume per cost dimension for this charging period.

2.16. ChargingProfileCriterionType

Class

A ChargingProfileCriterionType is a filter for charging profiles to be selected by a GetChargingProfilesRequest.

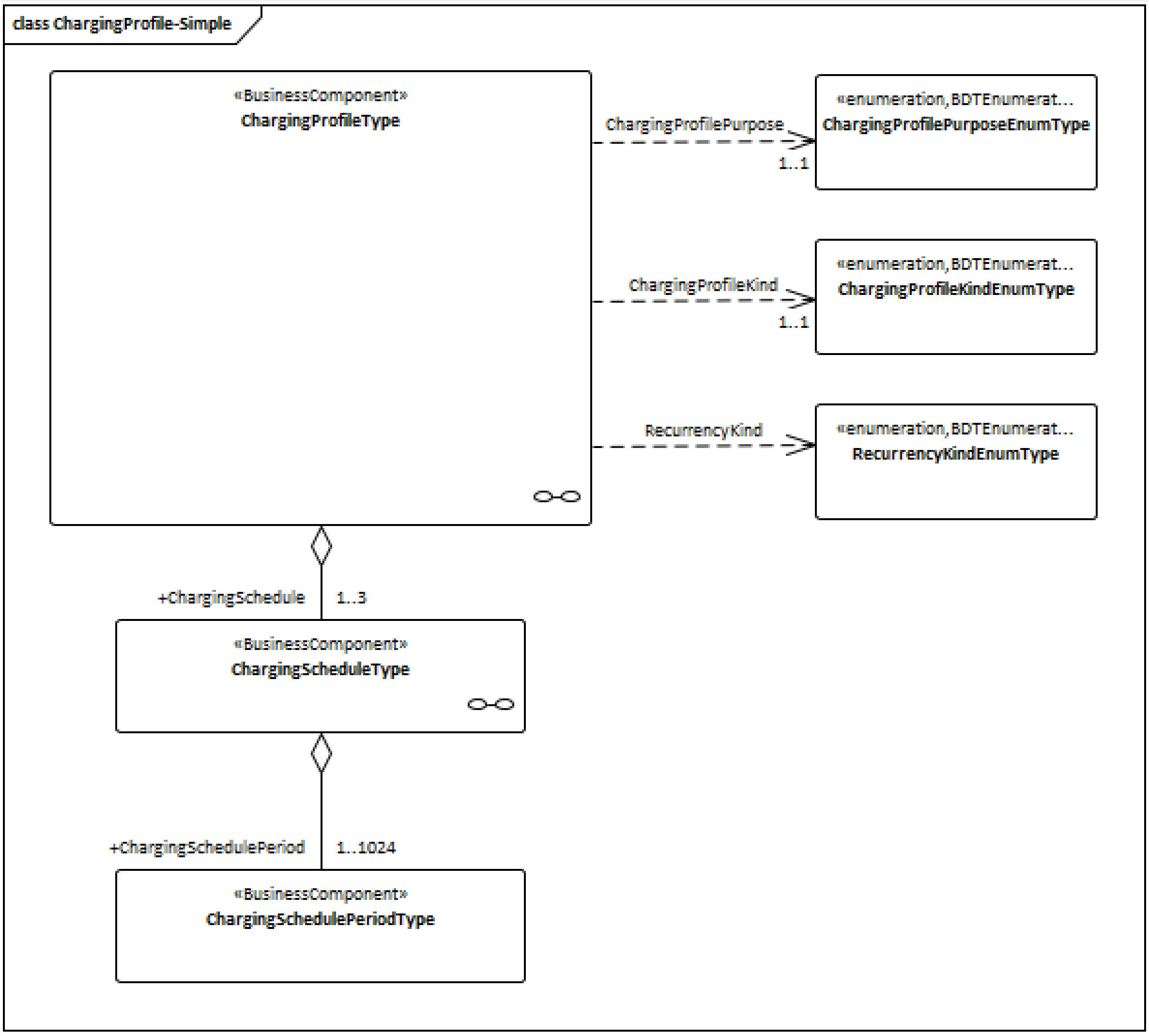
ChargingProfileCriterionType is used by: GetChargingProfilesRequest

Field Name	Field Type	Card.	Description
chargingProfilePurpose	ChargingProfilePurposeEnumType	0..1	Optional. Defines the purpose of the schedule transferred by this profile
stackLevel	integer, 0 < = val	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	string[0..20]	0..4	Optional. For which charging limit sources, charging profiles SHALL be reported. If omitted, the Charging Station SHALL not filter on chargingLimitSource. Values defined in Appendix as ChargingLimitSourceEnumStringType.

2.17. ChargingProfileType

Class

A ChargingProfile consists of 1 to 3 ChargingSchedules with a list of ChargingSchedulePeriods, 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. Unique within charging station. Id can have a negative value. This is useful to distinguish charging profiles from an external actor (external constraints) from charging profiles received from CSMS.
stackLevel	integer, 0 < = val	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.
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 if ChargingProfilePurpose is set to TxProfile in a SetChargingProfileRequest. The transactionId is used to match the profile to a specific transaction.

Field Name	Field Type	Card.	Description
maxOfflineDuration	integer	0..1	Optional. **(2.1)** Period in seconds that this charging profile remains valid after the Charging Station has gone offline. After this period the charging profile becomes invalid for as long as it is offline and the Charging Station reverts back to a valid profile with a lower stack level. If *invalidAfterOfflineDuration* is true, then this charging profile will become permanently invalid. A value of 0 means that the charging profile is immediately invalid while offline. When the field is absent, then no timeout applies and the charging profile remains valid when offline.
invalidAfterOfflineDuration	boolean	0..1	Optional. **(2.1)** When set to true this charging profile will not be valid anymore after being offline for more than *maxOfflineDuration*. When absent defaults to false.
dynUpdateInterval	integer	0..1	Optional. **(2.1)** Interval in seconds after receipt of last update, when to request a profile update by sending a PullDynamicScheduleUpdateRequest message. A value of 0 or no value means that no update interval applies. Only relevant in a dynamic charging profile.
dynUpdateTime	dateTime	0..1	Optional. **(2.1)** Time at which limits or setpoints in this charging profile were last updated by a PullDynamicScheduleUpdateRequest or UpdateDynamicScheduleRequest or by an external actor. Only relevant in a dynamic charging profile.
priceScheduleSignature	string[0..256]	0..1	Optional. **(2.1)** ISO 15118-20 signature for all price schedules in *chargingSchedules*. Note: for 256-bit elliptic curves (like secp256k1) the ECDSA signature is 512 bits (64 bytes) and for 521-bit curves (like secp521r1) the signature is 1042 bits. This equals 131 bytes, which can be encoded as base64 in 176 bytes.
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. Having multiple *chargingSchedules* is only allowed for charging profiles of purpose `TxProfile` in the context of an ISO 15118 charging session. For ISO 15118 Dynamic Control Mode only one chargingSchedule shall be provided.

2.18. ChargingSchedulePeriodType

Class

Charging schedule period structure defines a time period in a charging schedule. It is used in: CompositeScheduleType and in ChargingScheduleType. When used in a NotifyEVChargingScheduleRequest only *startPeriod*, *limit*, *limit_L2*, *limit_L3* are relevant.

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.

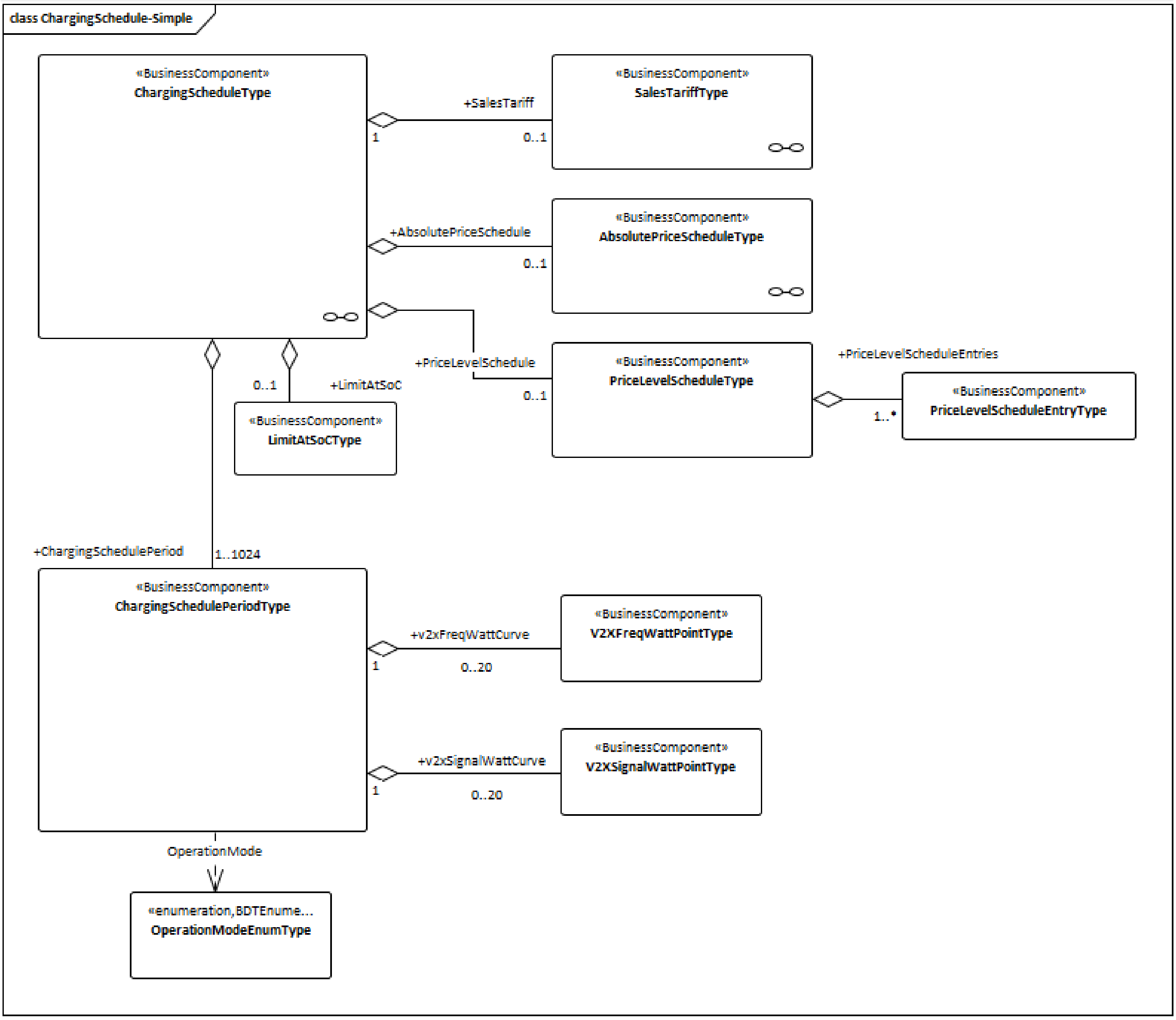
Field Name	Field Type	Card.	Description
limit	decimal	0..1	Optional. Optional only when not required by the *operationMode*, as in CentralSetpoint, ExternalSetpoint, ExternalLimits, LocalFrequency, LocalLoadBalancing. Charging rate limit during the schedule period, in the applicable *chargingRateUnit*. This SHOULD be a non-negative value; a negative value is only supported for backwards compatibility with older systems that use a negative value to specify a discharging limit. When using *chargingRateUnit* = `W`, this field represents the sum of the power of all phases, unless values are provided for L2 and L3, in which case this field represents phase L1.
limit_L2	decimal	0..1	Optional. **(2.1)** Charging rate limit on phase L2 in the applicable *chargingRateUnit*.
limit_L3	decimal	0..1	Optional. **(2.1)** Charging rate limit on phase L3 in the applicable *chargingRateUnit*.
numberPhases	integer, 0 < = val < = 3	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 < = val < = 3	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.
dischargeLimit	decimal, val < = 0	0..1	Optional. **(2.1)** Limit in *chargingRateUnit* that the EV is allowed to discharge with. Note, these are negative values in order to be consistent with *setpoint*, which can be positive and negative. For AC this field represents the sum of all phases, unless values are provided for L2 and L3, in which case this field represents phase L1.
dischargeLimit_L2	decimal, val < = 0	0..1	Optional. **(2.1)** Limit in *chargingRateUnit* on phase L2 that the EV is allowed to discharge with.
dischargeLimit_L3	decimal, val < = 0	0..1	Optional. **(2.1)** Limit in *chargingRateUnit* on phase L3 that the EV is allowed to discharge with.
setpoint	decimal	0..1	Optional. **(2.1)** Setpoint in *chargingRateUnit* that the EV should follow as close as possible. Use negative values for discharging. When a limit and/or *dischargeLimit* are given the overshoot when following *setpoint* must remain within these values. This field represents the sum of all phases, unless values are provided for L2 and L3, in which case this field represents phase L1.
setpoint_L2	decimal	0..1	Optional. **(2.1)** Setpoint in *chargingRateUnit* that the EV should follow on phase L2 as close as possible.
setpoint_L3	decimal	0..1	Optional. **(2.1)** Setpoint in *chargingRateUnit* that the EV should follow on phase L3 as close as possible.
setpointReactive	decimal	0..1	Optional. **(2.1)** Setpoint for reactive power (or current) in *chargingRateUnit* that the EV should follow as closely as possible. Positive values for inductive, negative for capacitive reactive power or current. This field represents the sum of all phases, unless values are provided for L2 and L3, in which case this field represents phase L1.
setpointReactive_L2	decimal	0..1	Optional. **(2.1)** Setpoint for reactive power (or current) in *chargingRateUnit* that the EV should follow on phase L2 as closely as possible.

Field Name	Field Type	Card.	Description
setpointReactive_L3	decimal	0..1	Optional. **(2.1)** Setpoint for reactive power (or current) in *chargingRateUnit* that the EV should follow on phase L3 as closely as possible.
preconditioningRequest	boolean	0..1	Optional. **(2.1)** If true, the EV should attempt to keep the BMS preconditioned for this time interval.
evseSleep	boolean	0..1	Optional. **(2.1)** If true, the EVSE must turn off power electronics/modules associated with this transaction. Default value when absent is false.
v2xBaseline	decimal	0..1	Optional. **(2.1)** Power value that, when present, is used as a baseline on top of which values from *v2xFreqWattCurve* and *v2xSignalWattCurve* are added.
operationMode	OperationModeEnumType	0..1	Optional. **(2.1)** Charging operation mode to use during this time interval. When absent defaults to `ChargingOnly`.
v2xFreqWattCurve	V2XFreqWattPointType	0..20	Optional. **(2.1)** Only required when *operationMode* = `LocalFrequency`. When used it must contain at least two coordinates to specify a power-frequency table to use during this period. The table determines the value of *setpoint* power for a given frequency. *chargingRateUnit* must be W for LocalFrequency control.
v2xSignalWattCurve	V2XSignalWattPointType	0..20	Optional. **(2.1)** Only used, but not required, when *operationMode* = `LocalFrequency`. When used it must contain at least two coordinates to specify a signal-frequency curve to use during this period. The curve determines the value of *setpoint* power for a given *signal*. *chargingRateUnit* must be `W` for `LocalFrequency` control.

2.19. ChargingScheduleType

Class

Charging schedule structure defines a list of charging periods, as used in: NotifyEVChargingScheduleRequest and ChargingProfileType. When used in a NotifyEVChargingScheduleRequest only *duration* and *chargingSchedulePeriod* are relevant and *chargingRateUnit* must be 'W'.
An ISO 15118-20 session may provide either an *absolutePriceSchedule* or a *priceLevelSchedule*. An ISO 15118-2 session can only provide a_salesTariff_ element. The field *digestValue* is used when price schedule or sales tariff are signed.



ChargingScheduleType is used by: Common:ChargingProfileType , NotifyChargingLimitRequest , NotifyEVChargingScheduleRequest

Field Name	Field Type	Card.	Description
id	integer	1..1	Required.
startSchedule	dateTime	0..1	Optional. Starting point of an absolute schedule or recurring schedule.
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 in case startSchedule is absent.
chargingRateUnit	ChargingRateUnitEnumType	1..1	Required. The unit of measure in which limits and setpoints are expressed.
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.
powerTolerance	decimal	0..1	Optional. **(2.1)** Power tolerance when following EVPowerProfile.
signatureId	integer, 0 < = val	0..1	Optional. **(2.1)** Id of this element for referencing in a signature.

Field Name	Field Type	Card.	Description
digestValue	string[0..88]	0..1	Optional. **(2.1)** Base64 encoded hash (SHA256 for ISO 15118-2, SHA512 for ISO 15118-20) of the EXI price schedule element. Used in signature.
useLocalTime	boolean	0..1	Optional. **(2.1)** Defaults to false. When true, disregard time zone offset in dateTime fields of *ChargingScheduleType* and use unqualified local time at Charging Station instead. This allows the same `Absolute` or `Recurring` charging profile to be used in both summer and winter time.
randomizedDelay	integer, 0 < = val	0..1	Optional. **(2.1)** Defaults to 0. When *randomizedDelay* not equals zero, then the start of each ChargingSchedulePeriodType is delayed by a randomly chosen number of seconds between 0 and *randomizedDelay*. Only allowed for TxProfile and TxDefaultProfile.
salesTariff	SalesTariffType	0..1	Optional. Sales tariff for charging associated with this schedule.
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.
absolutePriceSchedule	AbsolutePriceScheduleType	0..1	Optional. **(2.1)** The ISO 15118-20 absolute price schedule.
priceLevelSchedule	PriceLevelScheduleType	0..1	Optional. **(2.1)** The ISO 15118-20 price level schedule
limitAtSoC	LimitAtSoCType	0..1	Optional. **(2.1)** When present and SoC of EV is greater than or equal to *soc*, then charging limit or setpoint will be capped to the value of *limit*.

2.20. ChargingScheduleUpdateType

Class

Updates to a ChargingSchedulePeriodType for dynamic charging profiles.

ChargingScheduleUpdateType is used by: PullDynamicScheduleUpdateResponse , UpdateDynamicScheduleRequest

Field Name	Field Type	Card.	Description
limit	decimal	0..1	Optional. Optional only when not required by the *operationMode*, as in CentralSetpoint, ExternalSetpoint, ExternalLimits, LocalFrequency, LocalLoadBalancing. Charging rate limit during the schedule period, in the applicable *chargingRateUnit*. This SHOULD be a non-negative value; a negative value is only supported for backwards compatibility with older systems that use a negative value to specify a discharging limit. For AC this field represents the sum of all phases, unless values are provided for L2 and L3, in which case this field represents phase L1.
limit_L2	decimal	0..1	Optional. **(2.1)** Charging rate limit on phase L2 in the applicable *chargingRateUnit*.
limit_L3	decimal	0..1	Optional. **(2.1)** Charging rate limit on phase L3 in the applicable *chargingRateUnit*.
dischargeLimit	decimal, val < = 0	0..1	Optional. **(2.1)** Limit in *chargingRateUnit* that the EV is allowed to discharge with. Note, these are negative values in order to be consistent with *setpoint*, which can be positive and negative. For AC this field represents the sum of all phases, unless values are provided for L2 and L3, in which case this field represents phase L1.

Field Name	Field Type	Card.	Description
dischargeLimit_L2	decimal, val < = 0	0..1	Optional. **(2.1)** Limit in *chargingRateUnit* on phase L2 that the EV is allowed to discharge with.
dischargeLimit_L3	decimal, val < = 0	0..1	Optional. **(2.1)** Limit in *chargingRateUnit* on phase L3 that the EV is allowed to discharge with.
setpoint	decimal	0..1	Optional. **(2.1)** Setpoint in *chargingRateUnit* that the EV should follow as close as possible. Use negative values for discharging. When a limit and/or *dischargeLimit* are given the overshoot when following *setpoint* must remain within these values. This field represents the sum of all phases, unless values are provided for L2 and L3, in which case this field represents phase L1.
setpoint_L2	decimal	0..1	Optional. **(2.1)** Setpoint in *chargingRateUnit* that the EV should follow on phase L2 as close as possible.
setpoint_L3	decimal	0..1	Optional. **(2.1)** Setpoint in *chargingRateUnit* that the EV should follow on phase L3 as close as possible.
setpointReactive	decimal	0..1	Optional. **(2.1)** Setpoint for reactive power (or current) in *chargingRateUnit* that the EV should follow as closely as possible. Positive values for inductive, negative for capacitive reactive power or current. This field represents the sum of all phases, unless values are provided for L2 and L3, in which case this field represents phase L1.
setpointReactive_L2	decimal	0..1	Optional. **(2.1)** Setpoint for reactive power (or current) in *chargingRateUnit* that the EV should follow on phase L2 as closely as possible.
setpointReactive_L3	decimal	0..1	Optional. **(2.1)** Setpoint for reactive power (or current) in *chargingRateUnit* that the EV should follow on phase L3 as closely as possible.

2.21. 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.

2.22. ClearChargingProfileType

Class

A ClearChargingProfileType is a filter for charging profiles to be cleared by ClearChargingProfileRequest.

ClearChargingProfileType is used by: ClearChargingProfileRequest

Field Name	Field Type	Card.	Description
evseId	integer, 0 < = val	0..1	Optional. Specifies the id of the EVSE for which to clear charging profiles. An evseId 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 < = val	0..1	Optional. Specifies the stackLevel for which charging profiles will be cleared, if they meet the other criteria in the request.

2.23. 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, 0 < = val	1..1	Required. Id of the monitor of which a clear was requested.
statusInfo	StatusInfoType	0..1	Optional. Element providing more information about the status.

2.24. ClearTariffsResultType

Class

ClearTariffsResultType is used by: ClearTariffsResponse

Field Name	Field Type	Card.	Description
tariffId	string[0..60]	0..1	Optional. Id of tariff for which *status* is reported. If no tariffs were found, then this field is absent, and *status* will be `NoTariff`.
status	TariffClearStatusEnumType	1..1	Required.
statusInfo	StatusInfoType	0..1	Optional. Additional info on status

2.25. 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.

Field Name	Field Type	Card.	Description
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.

2.26. 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 for which the report is requested.

2.27. CompositeScheduleType

Class

CompositeScheduleType is used by: GetCompositeScheduleResponse

Field Name	Field Type	Card.	Description
evseId	integer, 0 < = val	1..1	Required.
duration	integer	1..1	Required.
scheduleStart	dateTime	1..1	Required.
chargingRateUnit	ChargingRateUnitEnumType	1..1	Required.
chargingSchedulePeriod	ChargingSchedulePeriodType	1..*	Required. List of ChargingSchedulePeriod elements defining maximum power or current over time.

2.28. ConstantStreamDataType

Class

ConstantStreamDataType is used by: OpenPeriodicEventStreamRequest , GetPeriodicEventStreamResponse

Field Name	Field Type	Card.	Description
id	integer, 0 < = val	1..1	Required. Uniquely identifies the stream
variableMonitoringId	integer, 0 < = val	1..1	Required. Id of monitor used to report his event. It can be a preconfigured or hardwired monitor.
params	PeriodicEventStreamParamsType	1..1	Required. Max time and items parameters

2.29. 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.

Field Name	Field Type	Card.	Description
cost	CostType	1..3	Required. This field contains the cost details.

2.30. CostDetailsType

Class

CostDetailsType contains the cost as calculated by Charging Station based on provided TariffType.

NOTE Reservation is not shown as a *chargingPeriod*, because it took place outside of the transaction.

CostDetailsType is used by: TransactionEventRequest

Field Name	Field Type	Card.	Description
failureToCalculate	boolean	0..1	Optional. If set to true, then Charging Station has failed to calculate the cost.
failureReason	string[0..500]	0..1	Optional. Optional human-readable reason text in case of failure to calculate.
chargingPeriods	ChargingPeriodType	0..*	Optional. List of Charging Periods that make up this charging session. A finished session has of 1 or more periods, where each period has a different list of *dimensions* that determined the price. When sent as a running cost update during a transaction *chargingPeriods* are omitted.
totalCost	TotalCostType	1..1	Required. Total sum of all the costs of this transaction in the specified currency.
totalUsage	TotalUsageType	1..1	Required. Total usage of energy and time

2.31. CostDimensionType

Class

Volume consumed of cost dimension.

CostDimensionType is used by: Common:ChargingPeriodType

Field Name	Field Type	Card.	Description
type	CostDimensionEnumType	1..1	Required. Type of cost dimension: energy, power, time, etc.
volume	decimal	1..1	Required. Volume of the dimension consumed, measured according to the dimension type.

2.32. 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

2.33. DCChargingParametersType

Class

EV DC charging parameters for ISO 15118-2

DCChargingParametersType is used by: Common:ChargingNeedsType

Field Name	Field Type	Card.	Description
evMaxCurrent	decimal	1..1	Required. Maximum current (in A) supported by the electric vehicle. Includes cable capacity. Relates to: **ISO 15118-2**: DC_EVChargeParameterType:EVMaximumCurrentLimit
evMaxVoltage	decimal	1..1	Required. Maximum voltage supported by the electric vehicle. Relates to: **ISO 15118-2**: DC_EVChargeParameterType: EVMaximumVoltageLimit
evMaxPower	decimal	0..1	Optional. Maximum power (in W) supported by the electric vehicle. Required for DC charging. Relates to: **ISO 15118-2**: DC_EVChargeParameterType: EVMaximumPowerLimit
evEnergyCapacity	decimal	0..1	Optional. Capacity of the electric vehicle battery (in Wh). Relates to: **ISO 15118-2**: DC_EVChargeParameterType: EVEnergyCapacity
energyAmount	decimal	0..1	Optional. Amount of energy requested (in Wh). This inludes energy required for preconditioning. Relates to: **ISO 15118-2**: DC_EVChargeParameterType: EVEnergyRequest
stateOfCharge	integer, 0 < = val < = 100	0..1	Optional. Energy available in the battery (in percent of the battery capacity) Relates to: **ISO 15118-2**: DC_EVChargeParameterType: DC_EVStatus: EVRESSSOC
fullSoC	integer, 0 < = val < = 100	0..1	Optional. Percentage of SoC at which the EV considers the battery fully charged. (possible values: 0 - 100) Relates to: **ISO 15118-2**: DC_EVChargeParameterType: FullSOC
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) Relates to: **ISO 15118-2**: DC_EVChargeParameterType: BulkSOC

2.34. DERChargingParametersType

Class

(2.1) DERChargingParametersType is used in ChargingNeedsType during an ISO 15118-20 session for AC_BPT_DER to report the inverter settings related to DER control that were agreed between EVSE and EV.

Fields starting with "ev" contain values from the EV. Other fields contain a value that is supported by both EV and EVSE.

DERChargingParametersType type is only relevant in case of an ISO 15118-20 AC_BPT_DER/AC_DER charging session.

NOTE All these fields have values greater or equal to zero (i.e. are non-negative)

DERChargingParametersType is used by: Common:ChargingNeedsType

Field Name	Field Type	Card.	Description
evSupportedDERControl	DERControlEnumType	0..*	Optional. DER control functions supported by EV. **ISO 15118-20**: DER_BPT_AC_CPDReqEnergyTransferModeType:DERControlFunctions (bitmap)
evOverExcitedMaxDischargePower	decimal	0..1	Optional. Rated maximum injected active power by EV, at specified over-excited power factor (overExcitedPowerFactor). It can also be defined as the rated maximum discharge power at the rated minimum injected reactive power value. This means that if the EV is providing reactive power support, and it is requested to discharge at max power (e.g. to satisfy an EMS request), the EV may override the request and discharge up to overExcitedMaximumDischargePower to meet the minimum reactive power requirements. Corresponds to the WOvPF attribute in IEC 61850. **ISO 15118-20**: DER_BPT_AC_CPDReqEnergyTransferModeType: EVOverExcitedMaximumDischargePower
evOverExcitedPowerFactor	decimal	0..1	Optional. EV power factor when injecting (over excited) the minimum reactive power. Corresponds to the OvPF attribute in IEC 61850. **ISO 15118-20**: DER_BPT_AC_CPDReqEnergyTransferModeType: EVOverExcitedPowerFactor
evUnderExcitedMaxDischargePower	decimal	0..1	Optional. Rated maximum injected active power by EV supported at specified under-excited power factor (EVUnderExcitedPowerFactor). It can also be defined as the rated maximum dischargePower at the rated minimum absorbed reactive power value. This means that if the EV is providing reactive power support, and it is requested to discharge at max power (e.g. to satisfy an EMS request), the EV may override the request and discharge up to underExcitedMaximumDischargePower to meet the minimum reactive power requirements. This corresponds to the WUnPF attribute in the IEC 61850. **ISO 15118-20**: DER_BPT_AC_CPDReqEnergyTransferModeType: EVUnderExcitedMaximumDischargePower
evUnderExcitedPowerFactor	decimal	0..1	Optional. EV power factor when injecting (under excited) the minimum reactive power. Corresponds to the OvPF attribute in IEC 61850. **ISO 15118-20**: DER_BPT_AC_CPDReqEnergyTransferModeType: EVUnderExcitedPowerFactor
maxApparentPower	decimal	0..1	Optional. Rated maximum total apparent power, defined by min(EV, EVSE) in va. Corresponds to the VAMaxRtg in IEC 61850. **ISO 15118-20**: DER_BPT_AC_CPDReqEnergyTransferModeType: EVMaximumApparentPower
maxChargeApparentPower	decimal	0..1	Optional. Rated maximum absorbed apparent power, defined by min(EV, EVSE) in va. This field represents the sum of all phases, unless values are provided for L2 and L3, in which case this field represents phase L1. Corresponds to the ChaVAMaxRtg in IEC 61850. **ISO 15118-20**: DER_BPT_AC_CPDReqEnergyTransferModeType: EVMaximumChargeApparentPower

Field Name	Field Type	Card.	Description
maxChargeApparentPower_L2	decimal	0..1	Optional. Rated maximum absorbed apparent power on phase L2, defined by min(EV, EVSE) in va. Corresponds to the ChaVAMaxRtg in IEC 61850. **ISO 15118-20**: DER_BPT_AC_CPDReqEnergyTransferModeType: EVMaximumChargeApparentPower_L2
maxChargeApparentPower_L3	decimal	0..1	Optional. Rated maximum absorbed apparent power on phase L3, defined by min(EV, EVSE) in va. Corresponds to the ChaVAMaxRtg in IEC 61850. **ISO 15118-20**: DER_BPT_AC_CPDReqEnergyTransferModeType: EVMaximumChargeApparentPower_L3
maxDischargeApparentPower	decimal	0..1	Optional. Rated maximum injected apparent power, defined by min(EV, EVSE) in va. This field represents the sum of all phases, unless values are provided for L2 and L3, in which case this field represents phase L1. Corresponds to the DisVAMaxRtg in IEC 61850. **ISO 15118-20**: DER_BPT_AC_CPDReqEnergyTransferModeType: EVMaximumDischargeApparentPower
maxDischargeApparentPower_L2	decimal	0..1	Optional. Rated maximum injected apparent power on phase L2, defined by min(EV, EVSE) in va. Corresponds to the DisVAMaxRtg in IEC 61850. **ISO 15118-20**: DER_BPT_AC_CPDReqEnergyTransferModeType: EVMaximumDischargeApparentPower_L2
maxDischargeApparentPower_L3	decimal	0..1	Optional. Rated maximum injected apparent power on phase L3, defined by min(EV, EVSE) in va. Corresponds to the DisVAMaxRtg in IEC 61850. **ISO 15118-20**: DER_BPT_AC_CPDReqEnergyTransferModeType: EVMaximumDischargeApparentPower_L3
maxChargeReactivePower	decimal	0..1	Optional. Rated maximum absorbed reactive power, defined by min(EV, EVSE), in vars. This field represents the sum of all phases, unless values are provided for L2 and L3, in which case this field represents phase L1. Corresponds to the AvarMax attribute in the IEC 61850. **ISO 15118-20**: DER_BPT_AC_CPDReqEnergyTransferModeType: EVMaximumChargeReactivePower
maxChargeReactivePower_L2	decimal	0..1	Optional. Rated maximum absorbed reactive power, defined by min(EV, EVSE), in vars on phase L2. Corresponds to the AvarMax attribute in the IEC 61850. **ISO 15118-20**: DER_BPT_AC_CPDReqEnergyTransferModeType: EVMaximumChargeReactivePower_L2
maxChargeReactivePower_L3	decimal	0..1	Optional. Rated maximum absorbed reactive power, defined by min(EV, EVSE), in vars on phase L3. Corresponds to the AvarMax attribute in the IEC 61850. **ISO 15118-20**: DER_BPT_AC_CPDReqEnergyTransferModeType: EVMaximumChargeReactivePower_L3

Field Name	Field Type	Card.	Description
minChargeReactivePower	decimal	0..1	Optional. Rated minimum absorbed reactive power, defined by max(EV, EVSE), in vars. This field represents the sum of all phases, unless values are provided for L2 and L3, in which case this field represents phase L1. **ISO 15118-20**: DER_BPT_AC_CPDReqEnergyTransferModeType: EVMinimumChargeReactivePower
minChargeReactivePower_L2	decimal	0..1	Optional. Rated minimum absorbed reactive power, defined by max(EV, EVSE), in vars on phase L2. **ISO 15118-20**: DER_BPT_AC_CPDReqEnergyTransferModeType: EVMinimumChargeReactivePower_L2
minChargeReactivePower_L3	decimal	0..1	Optional. Rated minimum absorbed reactive power, defined by max(EV, EVSE), in vars on phase L3. **ISO 15118-20**: DER_BPT_AC_CPDReqEnergyTransferModeType: EVMinimumChargeReactivePower_L3
maxDischargeReactivePower	decimal	0..1	Optional. Rated maximum injected reactive power, defined by min(EV, EVSE), in vars. This field represents the sum of all phases, unless values are provided for L2 and L3, in which case this field represents phase L1. Corresponds to the IvarMax attribute in the IEC 61850. **ISO 15118-20**: DER_BPT_AC_CPDReqEnergyTransferModeType: EVMaximumDischargeReactivePower
maxDischargeReactivePower_L2	decimal	0..1	Optional. Rated maximum injected reactive power, defined by min(EV, EVSE), in vars on phase L2. Corresponds to the IvarMax attribute in the IEC 61850. **ISO 15118-20**: DER_BPT_AC_CPDReqEnergyTransferModeType: EVMaximumDischargeReactivePower_L2
maxDischargeReactivePower_L3	decimal	0..1	Optional. Rated maximum injected reactive power, defined by min(EV, EVSE), in vars on phase L3. Corresponds to the IvarMax attribute in the IEC 61850. **ISO 15118-20**: DER_BPT_AC_CPDReqEnergyTransferModeType: EVMaximumDischargeReactivePower_L3
minDischargeReactivePower	decimal	0..1	Optional. Rated minimum injected reactive power, defined by max(EV, EVSE), in vars. This field represents the sum of all phases, unless values are provided for L2 and L3, in which case this field represents phase L1. **ISO 15118-20**: DER_BPT_AC_CPDReqEnergyTransferModeType: EVMinimumDischargeReactivePower
minDischargeReactivePower_L2	decimal	0..1	Optional. Rated minimum injected reactive power, defined by max(EV, EVSE), in var on phase L2. **ISO 15118-20**: DER_BPT_AC_CPDReqEnergyTransferModeType: EVMinimumDischargeReactivePower_L2
minDischargeReactivePower_L3	decimal	0..1	Optional. Rated minimum injected reactive power, defined by max(EV, EVSE), in var on phase L3. **ISO 15118-20**: DER_BPT_AC_CPDReqEnergyTransferModeType: EVMinimumDischargeReactivePower_L3
nominalVoltage	decimal	0..1	Optional. Line voltage supported by EVSE and EV. **ISO 15118-20**: DER_BPT_AC_CPDReqEnergyTransferModeType: EVNominalVoltage

Field Name	Field Type	Card.	Description
nominalVoltageOffset	decimal	0..1	Optional. The nominal AC voltage (rms) offset between the Charging Station’s electrical connection point and the utility’s point of common coupling. **ISO 15118-20**: DER_BPT_AC_CPDReqEnergyTransferModeType: EVNominalVoltageOffset
maxNominalVoltage	decimal	0..1	Optional. Maximum AC rms voltage, as defined by min(EV, EVSE) to operate with. **ISO 15118-20**: DER_BPT_AC_CPDReqEnergyTransferModeType: EVMaximumNominalVoltage
minNominalVoltage	decimal	0..1	Optional. Minimum AC rms voltage, as defined by max(EV, EVSE) to operate with. **ISO 15118-20**: DER_BPT_AC_CPDReqEnergyTransferModeType: EVMinimumNominalVoltage
evInverterManufacturer	string[0..50]	0..1	Optional. Manufacturer of the EV inverter. **ISO 15118-20**: DER_BPT_AC_CPDReqEnergyTransferModeType: EVInverterManufacturer
evInverterModel	string[0..50]	0..1	Optional. Model name of the EV inverter. **ISO 15118-20**: DER_BPT_AC_CPDReqEnergyTransferModeType: EVInverterModel
evInverterSerialNumber	string[0..50]	0..1	Optional. Serial number of the EV inverter. **ISO 15118-20**: DER_BPT_AC_CPDReqEnergyTransferModeType: EVInverterSerialNumber
evInverterSwVersion	string[0..50]	0..1	Optional. Software version of EV inverter. **ISO 15118-20**: DER_BPT_AC_CPDReqEnergyTransferModeType: EVInverterSwVersion
evInverterHwVersion	string[0..50]	0..1	Optional. Hardware version of EV inverter. **ISO 15118-20**: DER_BPT_AC_CPDReqEnergyTransferModeType: EVInverterHwVersion
evIslandingDetectionMethod	IslandingDetectionEnumType	0..*	Optional. Type of islanding detection method. Only mandatory when islanding detection is required at the site, as set in the ISO 15118 Service Details configuration. **ISO 15118-20**: DER_BPT_AC_CPDReqEnergyTransferModeType: EVIslandingDetectionMethod
evIslandingTripTime	decimal	0..1	Optional. Time after which EV will trip if an island has been detected. **ISO 15118-20**: DER_BPT_AC_CPDReqEnergyTransferModeType: EVIslandingTripTime
evMaximumLevel1DCInjection	decimal	0..1	Optional. Maximum injected DC current allowed at level 1 charging. **ISO 15118-20**: DER_BPT_AC_CPDReqEnergyTransferModeType: EVMaximumLevel1DCInjection
evDurationLevel1DCInjection	decimal	0..1	Optional. Maximum allowed duration of DC injection at level 1 charging. **ISO 15118-20**: DER_BPT_AC_CPDReqEnergyTransferModeType: EVDurationLevel1DCInjection

Field Name	Field Type	Card.	Description
evMaximumLevel2DCInjection	decimal	0..1	Optional. Maximum injected DC current allowed at level 2 charging. **ISO 15118-20**: DER_BPT_AC_CPDReqEnergyTransferModeType: EVMaximumLevel2DCInjection
evDurationLevel2DCInjection	decimal	0..1	Optional. Maximum allowed duration of DC injection at level 2 charging. **ISO 15118-20**: DER_BPT_AC_CPDReqEnergyTransferModeType: EVDurationLevel2DCInjection
evReactiveSusceptance	decimal	0..1	Optional. Measure of the susceptibility of the circuit to reactance, in Siemens (S). **ISO 15118-20**: DER_BPT_AC_CPDReqEnergyTransferModeType: EVReactiveSusceptance
evSessionTotalDischargeEnergyAvailable	decimal	0..1	Optional. Total energy value, in Wh, that EV is allowed to provide during the entire V2G session. The value is independent of the V2X Cycling area. Once this value reaches the value of 0, the EV may block any attempt to discharge in order to protect the battery health. **ISO 15118-20**: DER_BPT_AC_CPDReqEnergyTransferModeType: EVSessionTotalDischargeEnergyAvailable

2.35. DERCurveGetType

Class

DERCurveGetType is used by: ReportDERControlRequest

Field Name	Field Type	Card.	Description
id	identifierString[0..36]	1..1	Required. Id of DER curve
curveType	DERControlEnumType	1..1	Required. Type of DER curve
isDefault	boolean	1..1	Required. True if this is a default curve
isSuperseded	boolean	1..1	Required. True if this setting is superseded by a higher priority setting (i.e. lower value of *priority*)
curve	DERCurveType	1..1	Required. Parameters defining the DER curve

2.36. DERCurvePointsType

Class

DERCurvePointsType is used by: Common:DERCurveType

Field Name	Field Type	Card.	Description
x	decimal	1..1	Required. The data value of the X-axis (independent) variable, depending on the curve type.
y	decimal	1..1	Required. The data value of the Y-axis (dependent) variable, depending on the DERUnitEnumType of the curve. If *y* is power factor, then a positive value means DER is absorbing reactive power (under-excited), a negative value when DER is injecting reactive power (over-excited).

2.37. DERCurveType

Class

DERCurveType is used by: Common:DERCurveGetType , Common:LimitMaxDischargeType , SetDERControlRequest

Field Name	Field Type	Card.	Description
priority	integer, 0 < = val	1..1	Required. Priority of curve (0=highest)
yUnit	DERUnitEnumType	1..1	Required. Unit of the Y-axis of DER curve
responseTime	decimal	0..1	Optional. Open loop response time, the time to ramp up to 90% of the new target in response to the change in voltage, in seconds. A value of 0 is used to mean no limit. When not present, the device should follow its default behavior.
startTime	dateTime	0..1	Optional. Point in time when this curve will become activated. Only absent when *default* is true.
duration	decimal	0..1	Optional. Duration in seconds that this curve will be active. Only absent when *default* is true.
hysteresis	HysteresisType	0..1	Optional. Hysteresis parameters for curve.
voltageParams	VoltageParamsType	0..1	Optional. Additional parameters for voltage curves.
reactivePowerParams	ReactivePowerParamsType	0..1	Optional. Additional parameters for VoltVar curve.
curveData	DERCurvePointsType	1..10	Required. Coordinates of the DER curve. X-axis is determined by *curveType*. Y-axis is determined by *yUnit*.

2.38. EnterServiceGetType

Class

EnterServiceGetType is used by: ReportDERControlRequest

Field Name	Field Type	Card.	Description
id	identifierString[0..36]	1..1	Required. Id of setting
enterService	EnterServiceType	1..1	Required. Enter Service settings

2.39. EnterServiceType

Class

EnterServiceType is used by: Common:EnterServiceGetType , SetDERControlRequest

Field Name	Field Type	Card.	Description
priority	integer, 0 < = val	1..1	Required. Priority of setting (0=highest)
highVoltage	decimal	1..1	Required. Enter service voltage high
lowVoltage	decimal	1..1	Required. Enter service voltage low
highFreq	decimal	1..1	Required. Enter service frequency high
lowFreq	decimal	1..1	Required. Enter service frequency low
delay	decimal	0..1	Optional. Enter service delay
randomDelay	decimal	0..1	Optional. Enter service randomized delay
rampRate	decimal	0..1	Optional. Enter service ramp rate in seconds

2.40. EVAbsolutePriceScheduleEntryType

Class

(2.1) An entry in price schedule over time for which EV is willing to discharge.

EVAbsolutePriceScheduleEntryType is used by: Common:EVAbsolutePriceScheduleType

Field Name	Field Type	Card.	Description
duration	integer	1..1	Required. The amount of seconds of this entry.
evPriceRule	EVPriceRuleType	1..8	Required. A set of pricing rules for energy costs.

2.41. EVAbsolutePriceScheduleType

Class

(2.1) Price schedule of EV energy offer.

EVAbsolutePriceScheduleType is used by: Common:EVEnergyOfferType

Field Name	Field Type	Card.	Description
timeAnchor	dateTime	1..1	Required. Starting point in time of the EVEnergyOffer.
currency	string[0..3]	1..1	Required. Currency code according to ISO 4217.
priceAlgorithm	string[0..2000]	1..1	Required. ISO 15118-20 URN of price algorithm: Power, PeakPower, StackedEnergy.
evAbsolutePriceScheduleEntries	EVAbsolutePriceScheduleEntryType	1..1024	Required. Schedule of prices for which EV is willing to discharge.

2.42. EVEnergyOfferType

Class

(2.1) A schedule of the energy amount over time that EV is willing to discharge. A negative value indicates the willingness to discharge under specific conditions, a positive value indicates that the EV currently is not able to offer energy to discharge.

EVEnergyOfferType is used by: Common:ChargingNeedsType

Field Name	Field Type	Card.	Description
evPowerSchedule	EVPowerScheduleType	1..1	Required. Power schedule offered for discharging.
evAbsolutePriceSchedule	EVAbsolutePriceScheduleType	0..1	Optional. Price schedule for which EV is willing to discharge.

2.43. 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, 0 < = val	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 trigger for this event, e.g. exceeding a threshold value.
cause	integer, 0 < = val	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 (*attributeType* Actual) 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.

Field Name	Field Type	Card.	Description
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 < = val	0..1	Optional. Identifies the VariableMonitoring which triggered the event.
eventNotificationType	EventNotificationEnumType	1..1	Required. Specifies the event notification type of the message.
severity	integer, 0 < = val	0..1	Optional. **(2.1)** Severity associated with the monitor in *variableMonitoringId* or with the hardwired notification.
component	ComponentType	1..1	Required. Component for which event is notified.
variable	VariableType	1..1	Required. Variable for which event is notified.

2.44. EVPowerScheduleEntryType

Class

(2.1) An entry in schedule of the energy amount over time that EV is willing to discharge. A negative value indicates the willingness to discharge under specific conditions, a positive value indicates that the EV currently is not able to offer energy to discharge.

EVPowerScheduleEntryType is used by: Common:EVPowerScheduleType

Field Name	Field Type	Card.	Description
duration	integer	1..1	Required. The duration of this entry.
power	decimal	1..1	Required. Defines maximum amount of power for the duration of this EVPowerScheduleEntry to be discharged from the EV battery through EVSE power outlet. Negative values are used for discharging.

2.45. EVPowerScheduleType

Class

(2.1) Schedule of EV energy offer.

EVPowerScheduleType is used by: Common:EVEnergyOfferType

Field Name	Field Type	Card.	Description
timeAnchor	dateTime	1..1	Required. The time that defines the starting point for the EVEnergyOffer.
evPowerScheduleEntries	EVPowerScheduleEntryType	1..1024	Required. List of EVPowerScheduleEntries.

2.46. EVPriceRuleType

Class

(2.1) An entry in price schedule over time for which EV is willing to discharge.

EVPriceRuleType is used by: Common:EVAbsolutePriceScheduleEntryType

Field Name	Field Type	Card.	Description
energyFee	decimal	1..1	Required. Cost per kWh.

Field Name	Field Type	Card.	Description
powerRangeStart	decimal	1..1	Required. The EnergyFee applies between this value and the value of the PowerRangeStart of the subsequent EVPriceRule. If the power is below this value, the EnergyFee of the previous EVPriceRule applies. Negative values are used for discharging.

2.47. EVSEType

Class

Electric Vehicle Supply Equipment

EVSEType is used by: Common:ComponentType , TriggerMessageRequest , ChangeAvailabilityRequest , TransactionEventRequest

Field Name	Field Type	Card.	Description
id	integer, 0 < = val	1..1	Required. EVSE Identifier. This contains a number (> 0) designating an EVSE of the Charging Station.
connectorId	integer, 0 < = val	0..1	Optional. An id to designate a specific connector (on an EVSE) by connector index number.

2.48. 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..2000]	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.

2.49. FixedPFGetType

Class

FixedPFGetType is used by: ReportDERControlRequest

Field Name	Field Type	Card.	Description
id	identifierString[0..36]	1..1	Required. Id of setting.
isDefault	boolean	1..1	Required. True if setting is a default control.
isSuperseded	boolean	1..1	Required. True if this setting is superseded by a lower priority setting.
fixedPF	FixedPFType	1..1	Required. FixedPF for AbsorbW or InjectW

2.50. FixedPFType

Class

FixedPFType is used by: Common:FixedPFGetType , SetDERControlRequest

Field Name	Field Type	Card.	Description
priority	integer, 0 < = val	1..1	Required. Priority of setting (0=highest)
displacement	decimal	1..1	Required. Power factor, cos(phi), as value between 0..1.
excitation	boolean	1..1	Required. True when absorbing reactive power (under-excited), false when injecting reactive power (over-excited).
startTime	dateTime	0..1	Optional. Time when this setting becomes active
duration	decimal	0..1	Optional. Duration in seconds that this setting is active.

2.51. FixedVarGetType

Class

FixedVarGetType is used by: ReportDERControlRequest

Field Name	Field Type	Card.	Description
id	identifierString[0..36]	1..1	Required. Id of setting
isDefault	boolean	1..1	Required. True if setting is a default control.
isSuperseded	boolean	1..1	Required. True if this setting is superseded by a lower priority setting
fixedVar	FixedVarType	1..1	Required. Fixed Var setpoint

2.52. FixedVarType

Class

FixedVarType is used by: Common:FixedVarGetType , SetDERControlRequest

Field Name	Field Type	Card.	Description
priority	integer, 0 < = val	1..1	Required. Priority of setting (0=highest)
setpoint	decimal	1..1	Required. The value specifies a target var output interpreted as a signed percentage (-100 to 100). A negative value refers to charging, whereas a positive one refers to discharging. The value type is determined by the unit field.
unit	DERUnitEnumType	1..1	Required. Unit of the setpoint.
startTime	dateTime	0..1	Optional. Time when this setting becomes active.
duration	decimal	0..1	Optional. Duration in seconds that this setting is active.

2.53. FreqDroopGetType

Class

FreqDroopGetType is used by: ReportDERControlRequest

Field Name	Field Type	Card.	Description
id	identifierString[0..36]	1..1	Required. Id of setting
isDefault	boolean	1..1	Required. True if setting is a default control.
isSuperseded	boolean	1..1	Required. True if this setting is superseded by a higher priority setting (i.e. lower value of *priority*)
freqDroop	FreqDroopType	1..1	Required. FreqDroop parameters

2.54. FreqDroopType

Class

FreqDroopType is used by: Common:FreqDroopGetType , SetDERControlRequest

Field Name	Field Type	Card.	Description
priority	integer, 0 < = val	1..1	Required. Priority of setting (0=highest)
overFreq	decimal	1..1	Required. Over-frequency start of droop
underFreq	decimal	1..1	Required. Under-frequency start of droop
overDroop	decimal	1..1	Required. Over-frequency droop per unit, oFDroop
underDroop	decimal	1..1	Required. Under-frequency droop per unit, uFDroop
responseTime	decimal	1..1	Required. Open loop response time in seconds
startTime	dateTime	0..1	Optional. Time when this setting becomes active
duration	decimal	0..1	Optional. Duration in seconds that this setting is active

2.55. 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.

2.56. 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.
attributeType	AttributeEnumType	0..1	Optional.
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.
variable	VariableType	1..1	Required. Variable for which the attribute value is requested.
attributeStatusInfo	StatusInfoType	0..1	Optional. Detailed attribute status information.

2.57. GradientGetType

Class

GradientGetType is used by: ReportDERControlRequest

Field Name	Field Type	Card.	Description
id	identifierString[0..36]	1..1	Required. Id of setting
gradient	GradientType	1..1	Required. Gradient setting

2.58. GradientType

Class

GradientType is used by: Common:GradientGetType , SetDERControlRequest

Field Name	Field Type	Card.	Description
priority	integer, 0 < = val	1..1	Required. Id of setting
gradient	decimal	1..1	Required. Default ramp rate in seconds (0 if not applicable)
softGradient	decimal	1..1	Required. Soft-start ramp rate in seconds (0 if not applicable)

2.59. HysteresisType

Class

HysteresisType is used by: Common:DERCurveType

Field Name	Field Type	Card.	Description
hysteresisHigh	decimal	0..1	Optional. High value for return to normal operation after a grid event, in absolute value. This value adopts the same unit as defined by yUnit
hysteresisLow	decimal	0..1	Optional. Low value for return to normal operation after a grid event, in absolute value. This value adopts the same unit as defined by yUnit
hysteresisDelay	decimal	0..1	Optional. Delay in seconds, once grid parameter within HysteresisLow and HysteresisHigh, for the EV to return to normal operation after a grid event.
hysteresisGradient	decimal	0..1	Optional. Set default rate of change (ramp rate %/s) for the EV to return to normal operation after a grid event

2.60. 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].
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].

Field Name	Field Type	Card.	Description
evseId	integer, 0 < = val	0..*	Optional. Only used when the IdToken is only valid for one or more specific EVSEs, not for the entire Charging Station.
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.

2.61. 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 , BatterySwapRequest , RequestBatterySwapRequest

Field Name	Field Type	Card.	Description
idToken	identifierString[0..255]	1..1	Required. **(2.1)** IdToken is case insensitive. Might hold the hidden id of an RFID tag, but can for example also contain a UUID.
type	string[0..20]	1..1	Required. **(2.1)** Enumeration of possible idToken types. Values defined in Appendix as IdTokenEnumStringType.
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 *IdToken*. *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.

2.62. LimitAtSoCType

Class

LimitAtSoCType is used by: Common:ChargingScheduleType

Field Name	Field Type	Card.	Description
soc	integer, 0 < = val < = 100	1..1	Required. The SoC value beyond which the charging rate limit should be applied.
limit	decimal	1..1	Required. Charging rate limit beyond the SoC value. The unit is defined by *chargingSchedule.chargingRateUnit*.

2.63. LimitMaxDischargeGetType

Class

LimitMaxDischargeGetType is used by: ReportDERControlRequest

Field Name	Field Type	Card.	Description
id	identifierString[0..36]	1..1	Required. Id of setting
isDefault	boolean	1..1	Required. True if setting is a default control.
isSuperseded	boolean	1..1	Required. True if this setting is superseded by a higher priority setting (i.e. lower value of *priority*)

Field Name	Field Type	Card.	Description
limitMaxDischarge	LimitMaxDischargeType	1..1	Required. Maximum discharge power as percentage or rated capability

2.64. LimitMaxDischargeType

Class

LimitMaxDischargeType is used by: Common:LimitMaxDischargeGetType , SetDERControlRequest

Field Name	Field Type	Card.	Description
priority	integer, 0 < = val	1..1	Required. Priority of setting (0=highest)
pctMaxDischargePower	decimal	0..1	Optional. Only for PowerMonitoring. The value specifies a percentage (0 to 100) of the rated maximum discharge power of EV. The PowerMonitoring curve becomes active when power exceeds this percentage.
startTime	dateTime	0..1	Optional. Time when this setting becomes active
duration	decimal	0..1	Optional. Duration in seconds that this setting is active
powerMonitoringMustTrip	DERCurveType	0..1	Optional. The curve is an interpolation of data points where the x-axis values are time in seconds and the y-axis values refer to the percentage value of the rated EVMaximumDischargePower, reported in the ChargeParameterDiscoveryRequest message. The value lies between 0 and 100. The curve is activated when the power value measured via the ExternalMeter value reported in the ChargeLoopRes is higher than the pctMaxDischargePower defined above. If the power does not stay within the defined curve for the respective time period, the EV must trip.

2.65. LogParametersType

Class

Generic class for the configuration of logging entries.

LogParametersType is used by: GetLogRequest

Field Name	Field Type	Card.	Description
remoteLocation	string[0..2000]	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.

2.66. MessageContentType

Class

Contains message details, for a message to be displayed on a Charging Station.

MessageContentType is used by: Common:IdTokenInfoType , Common:MessageInfoType , Common:TariffType , TransactionEventResponse

Field Name	Field Type	Card.	Description
format	MessageFormatEnumType	1..1	Required. Format of the message.

Field Name	Field Type	Card.	Description
language	string[0..8]	0..1	Optional. Message language identifier. Contains a language code as defined in [RFC5646].
content	string[0..1024]	1..1	Required. **(2.1)** Required. Message contents. Maximum length supported by Charging Station is given in OCPPCommCtrlr.FieldLength["MessageContentType.content"]. Maximum length defaults to 1024.

2.67. 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, 0 < = val	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.
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.
messageExtra	MessageContentType	0..4	Optional. **(2.1)** Contains message details for extra languages to be displayed on a Charging Station.

2.68. 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

2.69. 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.

2.70. 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.

2.71. 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	0..1	Optional. **(2.1)** This field is ignored, since the OCPP version to use is determined during the websocket handshake. The field is only kept for backwards compatibility with the OCPP 2.0.1 JSON schema.
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.
ocppTransport	OCPPTransportEnumType	1..1	Required. Defines the transport protocol (e.g. SOAP or JSON). Note: SOAP is not supported in OCPP 2.x, but is supported by earlier versions of OCPP.
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.
ocppCsmsUrl	string[0..2000]	1..1	Required. URL of the CSMS(s) that this Charging Station communicates with, without the Charging Station identity part. The SecurityCtrlr.Identity field is appended to *ocppCsmsUrl* to provide the full websocket URL.
securityProfile	integer, 0 < = val	1..1	Required. This field specifies the security profile used when connecting to the CSMS with this NetworkConnectionProfile.
identity	string[0..48]	0..1	Optional. **(2.1)** Charging Station identity to be used as the basic authentication username.

Field Name	Field Type	Card.	Description
basicAuthPassword	string[0..64]	0..1	Optional. **(2.1)** BasicAuthPassword to use for security profile 1 or 2.
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.

2.72. OCSPRequestDataType

Class

Information about a certificate for an OCSP check.

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..2000]	1..1	Required. This contains the responder URL (Case insensitive).

2.73. OverstayRuleListType

Class

Part of ISO 15118-20 price schedule.

OverstayRuleListType is used by: Common:AbsolutePriceScheduleType

Field Name	Field Type	Card.	Description
overstayTimeThreshold	integer	0..1	Optional. Time till overstay is applied in seconds.
overstayPowerThreshold	RationalNumberType	0..1	Optional. Power threshold in W at which the overstay applies.
overstayRule	OverstayRuleType	1..5	Required. Overstay rules that will be applied.

2.74. OverstayRuleType

Class

Part of ISO 15118-20 price schedule.

OverstayRuleType is used by: Common:OverstayRuleListType

Field Name	Field Type	Card.	Description
overstayRuleDescription	string[0..32]	0..1	Optional. Human readable string to identify the overstay rule.
startTime	integer	1..1	Required. Time in seconds after trigger of the parent Overstay Rules for this particular fee to apply.
overstayFeePeriod	integer	1..1	Required. Time till overstay will be reapplied

Field Name	Field Type	Card.	Description
overstayFee	RationalNumberType	1..1	Required. Fee that applies to this overstay.

2.75. PeriodicEventStreamParamsType

Class

PeriodicEventStreamParamsType is used by: Common:ConstantStreamDataType , AdjustPeriodicEventStreamRequest , SetVariableMonitoringRequest.SetMonitoringDataType

Field Name	Field Type	Card.	Description
interval	integer, 0 < = val	0..1	Optional. Time in seconds after which stream data is sent.
values	integer, 0 < = val	0..1	Optional. Number of items to be sent together in stream.

2.76. PriceLevelScheduleEntryType

Class

Part of ISO 15118-20 price schedule.

PriceLevelScheduleEntryType is used by: Common:PriceLevelScheduleType

Field Name	Field Type	Card.	Description
duration	integer	1..1	Required. The amount of seconds that define the duration of this given PriceLevelScheduleEntry.
priceLevel	integer, 0 < = val	1..1	Required. Defines the price level of this PriceLevelScheduleEntry (referring to NumberOfPriceLevels). Small values for the PriceLevel represent a cheaper PriceLevelScheduleEntry. Large values for the PriceLevel represent a more expensive PriceLevelScheduleEntry.

2.77. PriceLevelScheduleType

Class

The PriceLevelScheduleType is modeled after the same type that is defined in ISO 15118-20, such that if it is supplied by an EMSP as a signed EXI message, the conversion from EXI to JSON (in OCPP) and back to EXI (for ISO 15118-20) does not change the digest and therefore does not invalidate the signature.

PriceLevelScheduleType is used by: Common:ChargingScheduleType

Field Name	Field Type	Card.	Description
timeAnchor	dateTime	1..1	Required. Starting point of this price schedule.
priceScheduleId	integer, 0 < = val	1..1	Required. Unique ID of this price schedule.
priceScheduleDescription	string[0..32]	0..1	Optional. Description of the price schedule.
numberOfPriceLevels	integer, 0 < = val	1..1	Required. Defines the overall number of distinct price level elements used across all PriceLevelSchedules.
priceLevelScheduleEntries	PriceLevelScheduleEntryType	1..100	Required. List of entries of the schedule.

2.78. PriceRuleStackType

Class

Part of ISO 15118-20 price schedule.

PriceRuleStackType is used by: Common:AbsolutePriceScheduleType

Field Name	Field Type	Card.	Description
duration	integer	1..1	Required. Duration of the stack of price rules. he amount of seconds that define the duration of the given PriceRule(s).
priceRule	PriceRuleType	1..8	Required. Contains the price rules.

2.79. PriceRuleType

Class

Part of ISO 15118-20 price schedule.

PriceRuleType is used by: Common:PriceRuleStackType

Field Name	Field Type	Card.	Description
parkingFeePeriod	integer	0..1	Optional. The duration of the parking fee period (in seconds). When the time enters into a ParkingFeePeriod, the ParkingFee will apply to the session. .
carbonDioxideEmission	integer, 0 < = val	0..1	Optional. Number of grams of CO2 per kWh.
renewableGenerationPercentage	integer, 0 < = val < = 100	0..1	Optional. Percentage of the power that is created by renewable resources.
energyFee	RationalNumberType	1..1	Required. Cost per kWh. Use zero for free energy.
parkingFee	RationalNumberType	0..1	Optional. Optional. Cost of parking. Mandatory whenever a parking fee applies.
powerRangeStart	RationalNumberType	1..1	Required. For values 0 and above, this is the power level above which this price rule applies. If there is another PriceRule with a higher value, and the current power is above that value, then that other PriceRule applies.For negative values, this is the power level below which this price rule applies. If there is another PriceRule with a lower value, and the current power is below that value, then that other PriceRule applies.

2.80. PriceType

Class

Price with and without tax. At least one of *exclTax*, *inclTax* must be present.

PriceType is used by: Common:TariffType , Common:TotalCostType

Field Name	Field Type	Card.	Description
exclTax	decimal	0..1	Optional. Price/cost excluding tax. Can be absent if *inclTax* is present.
inclTax	decimal	0..1	Optional. Price/cost including tax. Can be absent if *exclTax* is present.
taxRates	TaxRateType	0..5	Optional. Tax percentages that were used to calculate *inclTax* from *exclTax* (for displaying/printing on invoices).

2.81. RationalNumberType

Class

Part of ISO 15118-20 price schedule.

RationalNumberType is used by: Common:AbsolutePriceScheduleType , Common:AdditionalSelectedServicesType , Common:OverstayRuleListType , Common:OverstayRuleType , Common:PriceRuleType , Common:TaxRuleType

Field Name	Field Type	Card.	Description
exponent	integer	1..1	Required. The exponent to base 10 (dec)
value	integer	1..1	Required. Value which shall be multiplied.

2.82. ReactivePowerParamsType

Class

ReactivePowerParamsType is used by: Common:DERCurveType

Field Name	Field Type	Card.	Description
vRef	decimal	0..1	Optional. Only for VoltVar curve: The nominal ac voltage (rms) adjustment to the voltage curve points for Volt-Var curves (percentage).
autonomousVRefEnable	boolean	0..1	Optional. Only for VoltVar: Enable/disable autonomous VRef adjustment
autonomousVRefTimeConstant	decimal	0..1	Optional. Only for VoltVar: Adjustment range for VRef time constant

2.83. 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.

2.84. 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.

2.85. 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.

Field Name	Field Type	Card.	Description
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.

2.86. SalesTariffType

Class

A SalesTariff provided by a Mobility Operator (EMSP) . 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, 0 < = val	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 < = val	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.

2.87. 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.
measurand	MeasurandEnumType	0..1	Optional. Type of measurement. Default = "Energy.Active.Import.Register"
context	ReadingContextEnumType	0..1	Optional. Type of detail value: start, end or sample. Default = "Sample.Periodic"
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

2.88. SetMonitoringDataType

Class

Class to hold parameters of SetVariableMonitoring request.

SetMonitoringDataType is used by: SetVariableMonitoringRequest

Field Name	Field Type	Card.	Description
id	integer, 0 < = val	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.
severity	integer, 0 < = val	1..1	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. The severity levels have the following meaning: **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.
periodicEventStream	PeriodicEventStreamParamsType	0..1	Optional. **(2.1)** Optional. When present, events from a monitor will be sent via a periodic event stream. Used for monitors of type `Periodic`, `PeriodicClockAligned` or `Delta`.

2.89. SetMonitoringResultType

Class

Class to hold result of SetVariableMonitoring request.

SetMonitoringResultType is used by: SetVariableMonitoringResponse

Field Name	Field Type	Card.	Description
id	integer, 0 < = val	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.
severity	integer, 0 < = val	1..1	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. The severity levels have the following meaning: **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.

2.90. 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..2500]	1..1	Required. Value to be assigned to attribute of variable. This value is allowed to be an empty string (""). The Configuration Variable ConfigurationValueSize can be used to limit SetVariableData.attributeValue and VariableCharacteristics.valuesList. 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.

2.91. 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.

2.92. 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..32768]	1..1	Required. Base64 encoded, contains the signed data from the meter in the format specified in *encodingMethod*, which might contain more then just the meter value. It can contain information like timestamps, reference to a customer etc.
signingMethod	string[0..50]	0..1	Optional. **(2.1)** Method used to create the digital signature. Optional, if already included in *signedMeterData*. Standard values for this are defined in Appendix as SigningMethodEnumStringType.
encodingMethod	string[0..50]	1..1	Required. Format used by the energy meter to encode the meter data. For example: OCMF or EDL.
publicKey	string[0..2500]	0..1	Optional. **(2.1)** Base64 encoded, sending depends on configuration variable *PublicKeyWithSignedMeterValue*.

2.93. StatusInfoType

Class

Element providing more information about the status.

StatusInfoType is used by: Common:ClearTariffsResultType , BootNotificationResponse , CancelReservationResponse , TriggerMessageResponse , ChangeTransactionTariffResponse , GetTariffsResponse , OpenPeriodicEventStreamResponse , AdjustPeriodicEventStreamResponse , SetDERControlResponse , GetDERControlResponse , UnlockConnectorResponse , ClearDERControlResponse , UpdateFirmwareResponse , ClearDisplayMessageResponse , LogStatusNotificationRequest , Get15118EVCertificateResponse , GetCompositeScheduleResponse , ChangeAvailabilityResponse , GetLogResponse , ClearChargingProfileResponse , NotifyEVChargingNeedsResponse , ClearCacheResponse , NotifyEVChargingScheduleResponse , RequestStartTransactionResponse , RequestStopTransactionResponse , SetChargingProfileResponse , SetDisplayMessageResponse , SetNetworkProfileResponse , SignCertificateResponse , DataTransferResponse , FirmwareStatusNotificationRequest , CertificateSignedResponse , DeleteCertificateResponse , GetChargingProfilesResponse , GetInstalledCertificateIdsResponse , InstallCertificateResponse , ClearVariableMonitoringResponse.ClearMonitoringResultType , GetBaseReportResponse , GetMonitoringReportResponse , GetReportResponse , GetVariablesResponse.GetVariableResultType , ReserveNowResponse , SetMonitoringBaseResponse , SetMonitoringLevelResponse , SetVariableMonitoringResponse.SetMonitoringResultType , SetVariablesResponse.SetVariableResultType , PublishFirmwareResponse , PublishFirmwareStatusNotificationRequest , GetCertificateStatusResponse , ResetResponse , GetDisplayMessagesResponse , CustomerInformationResponse , NotifyAllowedEnergyTransferResponse , UsePriorityChargingResponse , PullDynamicScheduleUpdateResponse , UpdateDynamicScheduleResponse , SendLocalListResponse , AFRRSignalResponse , RequestBatterySwapResponse , VatNumberValidationResponse , SetDefaultTariffResponse

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..1024]	0..1	Optional. Additional text to provide detailed information.

2.94. StreamDataElementType

Class

StreamDataElementType is used by: NotifyPeriodicEventStream

Field Name	Field Type	Card.	Description
t	decimal	1..1	Required. Offset relative to *basetime* of this message. *basetime* + *t* is timestamp of recorded value.
v	string[0..2500]	1..1	Required.

2.95. TariffAssignmentType

Class

Shows assignment of tariffs to EVSE or IdToken.

TariffAssignmentType is used by: GetTariffsResponse

Field Name	Field Type	Card.	Description
tariffId	string[0..60]	1..1	Required. Tariff id.
tariffKind	TariffKindEnumType	1..1	Required. Kind of tariff (driver/default)
validFrom	dateTime	0..1	Optional. Date/time when this tariff become active.
evseIds	integer, 0 < = val	0..*	Optional.
idTokens	identifierString[0..255]	0..*	Optional. IdTokens related to tariff

2.96. TariffConditionsFixedType

Class

These conditions describe if a FixedPrice applies at start of the transaction.

When more than one restriction is set, they are to be treated as a logical AND. All need to be valid before this price is active.

NOTE *startTimeOfDay* and *endTimeOfDay* are in local time, because it is the time in the tariff as it is shown to the EV driver at the Charging Station. A Charging Station will convert this to the internal time zone that it uses (which is recommended to be UTC, see section Generic chapter 3.1) when performing cost calculation.

TariffConditionsFixedType is used by: Common:TariffFixedPriceType

Field Name	Field Type	Card.	Description
startTimeOfDay	string	0..1	Optional. Start time of day in local time. Format as per RFC 3339: time-hour ":" time-minute Must be in 24h format with leading zeros. Hour/Minute separator: ":" Regex: ([0-1][0-9]|2[0-3]):[0-5][0-9]
endTimeOfDay	string	0..1	Optional. End time of day in local time. Same syntax as *startTimeOfDay*. If end time < start time then the period wraps around to the next day. To stop at end of the day use: 00:00.
dayOfWeek	DayOfWeekEnumType	0..7	Optional. Day(s) of the week this is tariff applies.
validFromDate	string	0..1	Optional. Start date in local time, for example: 2015-12-24. Valid from this day (inclusive). Format as per RFC 3339: full-date Regex: ([12][0-9]{3})-(0[1-9]|1[0-2])-(0[1-9]|[12][0-9]|3[01])
validToDate	string	0..1	Optional. End date in local time, for example: 2015-12-27. Valid until this day (exclusive). Same syntax as *validFromDate*.
evseKind	EvseKindEnumType	0..1	Optional. Type of EVSE (AC, DC) this tariff applies to.
paymentBrand	string[0..20]	0..1	Optional. For which payment brand this (adhoc) tariff applies. Can be used to add a surcharge for certain payment brands. Based on value of *additionalIdToken* from *idToken.additionalInfo.type* = "PaymentBrand".
paymentRecognition	string[0..20]	0..1	Optional. Type of adhoc payment, e.g. CC, Debit. Based on value of *additionalIdToken* from *idToken.additionalInfo.type* = "PaymentRecognition".

2.97. TariffConditionsType

Class

These conditions describe if and when a TariffEnergyType or TariffTimeType applies during a transaction.

When more than one restriction is set, they are to be treated as a logical AND. All need to be valid before this price is active.

For reverse energy flow (discharging) negative values of energy, power and current are used.

NOTE *minXXX* (where XXX = Kwh/A/Kw) must be read as "closest to zero", and *maxXXX* as "furthest from zero". For example, a **charging** power range from 10 kW to 50 kWh is given by *minPower* = 10000 and *maxPower* = 50000, and a **discharging** power range from -10 kW to -50 kW is given by *minPower* = -10 and *maxPower* = -50.

NOTE *startTimeOfDay* and *endTimeOfDay* are in local time, because it is the time in the tariff as it is shown to the EV driver at the Charging Station. A Charging Station will convert this to the internal time zone that it uses (which is recommended to be UTC, see section Generic chapter 3.1) when performing cost calculation.

TariffConditionsType is used by: Common:TariffEnergyPriceType , Common:TariffTimePriceType

Field Name	Field Type	Card.	Description
startTimeOfDay	string	0..1	Optional. Start time of day in local time. Format as per RFC 3339: time-hour ":" time-minute Must be in 24h format with leading zeros. Hour/Minute separator: ":" Regex: ([0-1][0-9]|2[0-3]):[0-5][0-9]
endTimeOfDay	string	0..1	Optional. End time of day in local time. Same syntax as *startTimeOfDay*. If end time < start time then the period wraps around to the next day. To stop at end of the day use: 00:00.
dayOfWeek	DayOfWeekEnumType	0..7	Optional. Day(s) of the week this is tariff applies.
validFromDate	string	0..1	Optional. Start date in local time, for example: 2015-12-24. Valid from this day (inclusive). Format as per RFC 3339: full-date Regex: ([12][0-9]{3})-(0[1-9]|1[0-2])-(0[1-9]|[12][0-9]|3[01])
validToDate	string	0..1	Optional. End date in local time, for example: 2015-12-27. Valid until this day (exclusive). Same syntax as *validFromDate*.
evseKind	EvseKindEnumType	0..1	Optional. Type of EVSE (AC, DC) this tariff applies to.
minEnergy	decimal	0..1	Optional. Minimum consumed energy in Wh, for example 20000 Wh. Valid from this amount of energy (inclusive) being used.
maxEnergy	decimal	0..1	Optional. Maximum consumed energy in Wh, for example 50000 Wh. Valid until this amount of energy (exclusive) being used.
minCurrent	decimal	0..1	Optional. Sum of the minimum current (in Amperes) over all phases, for example 5 A. When the EV is charging with more than, or equal to, the defined amount of current, this price is/becomes active. If the charging current is or becomes lower, this price is not or no longer valid and becomes inactive. This is NOT about the minimum current over the entire transaction.
maxCurrent	decimal	0..1	Optional. Sum of the maximum current (in Amperes) over all phases, for example 20 A. When the EV is charging with less than the defined amount of current, this price becomes/is active. If the charging current is or becomes higher, this price is not or no longer valid and becomes inactive. This is NOT about the maximum current over the entire transaction.
minPower	decimal	0..1	Optional. Minimum power in W, for example 5000 W. When the EV is charging with more than, or equal to, the defined amount of power, this price is/becomes active. If the charging power is or becomes lower, this price is not or no longer valid and becomes inactive. This is NOT about the minimum power over the entire transaction.
maxPower	decimal	0..1	Optional. Maximum power in W, for example 20000 W. When the EV is charging with less than the defined amount of power, this price becomes/is active. If the charging power is or becomes higher, this price is not or no longer valid and becomes inactive. This is NOT about the maximum power over the entire transaction.
minTime	integer	0..1	Optional. Minimum duration in seconds the transaction (charging & idle) MUST last (inclusive). When the duration of a transaction is longer than the defined value, this price is or becomes active. Before that moment, this price is not yet active.

Field Name	Field Type	Card.	Description
maxTime	integer	0..1	Optional. Maximum duration in seconds the transaction (charging & idle) MUST last (exclusive). When the duration of a transaction is shorter than the defined value, this price is or becomes active. After that moment, this price is no longer active.
minChargingTime	integer	0..1	Optional. Minimum duration in seconds the charging MUST last (inclusive). When the duration of a charging is longer than the defined value, this price is or becomes active. Before that moment, this price is not yet active.
maxChargingTime	integer	0..1	Optional. Maximum duration in seconds the charging MUST last (exclusive). When the duration of a charging is shorter than the defined value, this price is or becomes active. After that moment, this price is no longer active.
minIdleTime	integer	0..1	Optional. Minimum duration in seconds the idle period (i.e. not charging) MUST last (inclusive). When the duration of the idle time is longer than the defined value, this price is or becomes active. Before that moment, this price is not yet active.
maxIdleTime	integer	0..1	Optional. Maximum duration in seconds the idle period (i.e. not charging) MUST last (exclusive). When the duration of idle time is shorter than the defined value, this price is or becomes active. After that moment, this price is no longer active.

2.98. TariffEnergyPriceType

Class

Tariff with optional conditions for an energy price.

TariffEnergyPriceType is used by: Common:TariffEnergyType

Field Name	Field Type	Card.	Description
priceKwh	decimal	1..1	Required. Price per kWh (excl. tax) for this element.
conditions	TariffConditionsType	0..1	Optional. Conditions when this tariff element price is applicable. When absent always applicable,

2.99. TariffEnergyType

Class

Price elements and tax for energy

TariffEnergyType is used by: Common:TariffType

Field Name	Field Type	Card.	Description
taxRates	TaxRateType	0..5	Optional. Applicable tax percentages for this tariff dimension. If omitted, no tax is applicable. Not providing a tax is different from 0% tax, which would be a value of 0.0 here.
prices	TariffEnergyPriceType	1..*	Required. Element tariff price and conditions

2.100. TariffFixedPriceType

Class

Tariff with optional conditions for a fixed price.

TariffFixedPriceType is used by: Common:TariffFixedType

Field Name	Field Type	Card.	Description
priceFixed	decimal	1..1	Required. Fixed price for this element e.g. a start fee.
conditions	TariffConditionsFixedType	0..1	Optional. Conditions when this tariff element price is applicable. When absent always applicable,

2.101. TariffFixedType

Class

TariffFixedType is used by: Common:TariffType

Field Name	Field Type	Card.	Description
prices	TariffFixedPriceType	1..*	Required.
taxRates	TaxRateType	0..5	Optional. Applicable tax percentages for this tariff dimension. If omitted, no tax is applicable. Not providing a tax is different from 0% tax, which would be a value of 0.0 here.

2.102. TariffTimePriceType

Class

Tariff with optional conditions for a time duration price.

TariffTimePriceType is used by: Common:TariffTimeType

Field Name	Field Type	Card.	Description
priceMinute	decimal	1..1	Required. Price per minute (excl. tax) for this element.
conditions	TariffConditionsType	0..1	Optional. Conditions when this tariff element price is applicable. When absent always applicable,

2.103. TariffTimeType

Class

Price elements and tax for time

TariffTimeType is used by: Common:TariffType

Field Name	Field Type	Card.	Description
prices	TariffTimePriceType	1..*	Required. Element tariff price and conditions
taxRates	TaxRateType	0..5	Optional. Applicable tax percentages for this tariff dimension. If omitted, no tax is applicable. Not providing a tax is different from 0% tax, which would be a value of 0.0 here.

2.104. TariffType

Class

A tariff is described by fields with prices for: energy, charging time, idle time, fixed fee, reservation time, reservation fixed fee.
Each of these fields may have (optional) conditions that specify when a price is applicable.
The *description* contains a human-readable explanation of the tariff to be shown to the user.
The other fields are parameters that define the tariff. These are used by the charging station to calculate the price.

TariffType is used by: ChangeTransactionTariffRequest , AuthorizeResponse , SetDefaultTariffRequest

Field Name	Field Type	Card.	Description
tariffId	string[0..60]	1..1	Required. Unique id of tariff
currency	string[0..3]	1..1	Required. Currency code according to ISO 4217
validFrom	dateTime	0..1	Optional. Time when this tariff becomes active. When absent, it is immediately active.
description	MessageContentType	0..10	Optional. List of multi-language tariff information texts to be shown to the user.
energy	TariffEnergyType	0..1	Optional. Energy tariff
chargingTime	TariffTimeType	0..1	Optional. Charging time tariff
idleTime	TariffTimeType	0..1	Optional. Idle time tariff
fixedFee	TariffFixedType	0..1	Optional. Fixed fee tariff
minCost	PriceType	0..1	Optional. The minimal cost for a transaction with this tariff including and excluding taxes. Minimum can be including tax or excluding tax, or both.
maxCost	PriceType	0..1	Optional. The maximum cost for a transaction with this tariff. Maximum can be including tax or excluding tax, or both.
reservationTime	TariffTimeType	0..1	Optional. Reservation time tariff
reservationFixed	TariffFixedType	0..1	Optional. Fixed fee for a reservation

2.105. TaxRateType

Class

Tax percentage

TaxRateType is used by: Common:PriceType , Common:TariffEnergyType , Common:TariffFixedType , Common:TariffTimeType

Field Name	Field Type	Card.	Description
type	string[0..20]	1..1	Required. Type of this tax, e.g. "Federal ", "State", for information on receipt.
tax	decimal	1..1	Required. Tax percentage
stack	integer, 0 < = val	0..1	Optional. Stack level for this type of tax. Default value, when absent, is 0. *stack* = 0: tax on net price; *stack* = 1: tax added on top of *stack* 0; *stack* = 2: tax added on top of *stack* 1, etc.

2.106. TaxRuleType

Class

Part of ISO 15118-20 price schedule.

TaxRuleType is used by: Common:AbsolutePriceScheduleType

Field Name	Field Type	Card.	Description
taxRuleID	integer, 0 < = val	1..1	Required. Id for the tax rule.
taxRuleName	string[0..100]	0..1	Optional. Human readable string to identify the tax rule.
taxIncludedInPrice	boolean	0..1	Optional. Indicates whether the tax is included in any price or not.
appliesToEnergyFee	boolean	1..1	Required. Indicates whether this tax applies to Energy Fees.
appliesToParkingFee	boolean	1..1	Required. Indicates whether this tax applies to Parking Fees.
appliesToOverstayFee	boolean	1..1	Required. Indicates whether this tax applies to Overstay Fees.

Field Name	Field Type	Card.	Description
appliesToMinimumMaximumCost	boolean	1..1	Required. Indicates whether this tax applies to Minimum/Maximum Cost.
taxRate	RationalNumberType	1..1	Required. Percentage of the total amount of applying fee (energy, parking, overstay, MinimumCost and/or MaximumCost).

2.107. TotalCostType

Class

This contains the cost calculated during a transaction. It is used both for running cost and final cost of the transaction.

TotalCostType is used by: Common:CostDetailsType

Field Name	Field Type	Card.	Description
currency	string[0..3]	1..1	Required. Currency of the costs in ISO 4217 Code.
typeOfCost	TariffCostEnumType	1..1	Required. Type of cost: normal or the minimum or maximum cost.
fixed	PriceType	0..1	Optional. Total sum of all flat fees in the specified currency, except for TariffFixedPriceTypes with *conditions.isReservation* = true (counted in *reservation*).
energy	PriceType	0..1	Optional. Total sum of all the cost of all the energy used, in the specified currency.
chargingTime	PriceType	0..1	Optional. Total sum of all the cost related to duration of charging during this transaction, in the specified currency.
idleTime	PriceType	0..1	Optional. Total sum of all the cost related to idle time of this transaction, including fixed price components, in the specified currency.
reservationTime	PriceType	0..1	Optional. Sum of all time-based cost related to reservation, i.e. TariffType.reservationTime, in the specified currency.
total	TotalPriceType	1..1	Required. Total of associated cost elements for fixed, energy, chargingTime, idleTime and reservation.
reservationFixed	PriceType	0..1	Optional. Sum of fixed cost related to reservation, i.e. TariffType.reservationFixed, in the specified currency.

2.108. TotalPriceType

Class

Total cost with and without tax. Contains the total of energy, charging time, idle time, fixed and reservation costs including and/or excluding tax.

TotalPriceType is used by: Common:TotalCostType

Field Name	Field Type	Card.	Description
exclTax	decimal	0..1	Optional. Price/cost excluding tax. Can be absent if *inclTax* is present.
inclTax	decimal	0..1	Optional. Price/cost including tax. Can be absent if *exclTax* is present.

2.109. TotalUsageType

Class

This contains the calculated usage of energy, charging time and idle time during a transaction.

TotalUsageType is used by: Common:CostDetailsType

Field Name	Field Type	Card.	Description
energy	decimal	1..1	Required.
chargingTime	integer	1..1	Required. Total duration of the charging session (including the duration of charging and not charging), in seconds.
idleTime	integer	1..1	Required. Total duration of the charging session where the EV was not charging (no energy was transferred between EVSE and EV), in seconds.
reservationTime	integer	0..1	Optional. Total time of reservation in seconds.

2.110. TransactionLimitType

Class

Cost, energy, time or SoC limit for a transaction.

TransactionLimitType is used by: Common:TransactionType , TransactionEventResponse

Field Name	Field Type	Card.	Description
maxCost	decimal	0..1	Optional. Maximum allowed cost of transaction in currency of tariff.
maxEnergy	decimal	0..1	Optional. Maximum allowed energy in Wh to charge in transaction.
maxTime	integer	0..1	Optional. Maximum duration of transaction in seconds from start to end.
maxSoC	integer, 0 < = val < = 100	0..1	Optional. Maximum State of Charge of EV in percentage.

2.111. 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. Omitted when there is no communication between EVSE and EV, because no cable is plugged in.
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. The *stoppedReason* 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 *stoppedReason* 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.
operationMode	OperationModeEnumType	0..1	Optional. **(2.1)** The *operationMode* that is currently in effect for the transaction.
tariffId	string[0..60]	0..1	Optional. **(2.1)** Id of tariff in use for transaction

Field Name	Field Type	Card.	Description
transactionLimit	TransactionLimitType	0..1	Optional. **(2.1)** Maximum cost/energy/time allowed for this transaction.

2.112. 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. The *multiplier* only multiplies the value of the measurand. It does not specify a conversion between units, for example, kW and W.

2.113. V2XChargingParametersType

Class

Charging parameters for ISO 15118-20, also supporting V2X charging/discharging.+ All values are greater or equal to zero, with the exception of EVMinEnergyRequest, EVMaxEnergyRequest, EVTargetEnergyRequest, EVMinV2XEnergyRequest and EVMaxV2XEnergyRequest.

V2XChargingParametersType is used by: Common:ChargingNeedsType

Field Name	Field Type	Card.	Description
minChargePower	decimal	0..1	Optional. Minimum charge power in W, defined by max(EV, EVSE). This field represents the sum of all phases, unless values are provided for L2 and L3, in which case this field represents phase L1. Relates to: **ISO 15118-20**: BPT_AC/DC_CPDReqEnergyTransferModeType: EVMinimumChargePower
minChargePower_L2	decimal	0..1	Optional. Minimum charge power on phase L2 in W, defined by max(EV, EVSE). Relates to: **ISO 15118-20**: BPT_AC/DC_CPDReqEnergyTransferModeType: EVMinimumChargePower_L2
minChargePower_L3	decimal	0..1	Optional. Minimum charge power on phase L3 in W, defined by max(EV, EVSE). Relates to: **ISO 15118-20**: BPT_AC/DC_CPDReqEnergyTransferModeType: EVMinimumChargePower_L3

Field Name	Field Type	Card.	Description
maxChargePower	decimal	0..1	Optional. Maximum charge (absorbed) power in W, defined by min(EV, EVSE) at unity power factor. This field represents the sum of all phases, unless values are provided for L2 and L3, in which case this field represents phase L1. It corresponds to the ChaWMax attribute in the IEC 61850. It is usually equivalent to the rated apparent power of the EV when discharging (ChaVAMax) in IEC 61850. Relates to: **ISO 15118-20**: BPT_AC/DC_CPDReqEnergyTransferModeType: EVMaximumChargePower
maxChargePower_L2	decimal	0..1	Optional. Maximum charge power on phase L2 in W, defined by min(EV, EVSE) Relates to: **ISO 15118-20**: BPT_AC/DC_CPDReqEnergyTransferModeType: EVMaximumChargePower_L2
maxChargePower_L3	decimal	0..1	Optional. Maximum charge power on phase L3 in W, defined by min(EV, EVSE) Relates to: **ISO 15118-20**: BPT_AC/DC_CPDReqEnergyTransferModeType: EVMaximumChargePower_L3
minDischargePower	decimal	0..1	Optional. Minimum discharge (injected) power in W, defined by max(EV, EVSE) at unity power factor. Value >= 0. This field represents the sum of all phases, unless values are provided for L2 and L3, in which case this field represents phase L1. It corresponds to the WMax attribute in the IEC 61850. It is usually equivalent to the rated apparent power of the EV when discharging (VAMax attribute in the IEC 61850). Relates to: **ISO 15118-20**: BPT_AC/DC_CPDReqEnergyTransferModeType: EVMinimumDischargePower
minDischargePower_L2	decimal	0..1	Optional. Minimum discharge power on phase L2 in W, defined by max(EV, EVSE). Value >= 0. Relates to: **ISO 15118-20**: BPT_AC/DC_CPDReqEnergyTransferModeType: EVMinimumDischargePower_L2
minDischargePower_L3	decimal	0..1	Optional. Minimum discharge power on phase L3 in W, defined by max(EV, EVSE). Value >= 0. Relates to: **ISO 15118-20**: BPT_AC/DC_CPDReqEnergyTransferModeType: EVMinimumDischargePower_L3
maxDischargePower	decimal	0..1	Optional. Maximum discharge (injected) power in W, defined by min(EV, EVSE) at unity power factor. Value >= 0. This field represents the sum of all phases, unless values are provided for L2 and L3, in which case this field represents phase L1. Relates to: **ISO 15118-20**: BPT_AC/DC_CPDReqEnergyTransferModeType: EVMaximumDischargePower
maxDischargePower_L2	decimal	0..1	Optional. Maximum discharge power on phase L2 in W, defined by min(EV, EVSE). Value >= 0. Relates to: **ISO 15118-20**: BPT_AC/DC_CPDReqEnergyTransferModeType: EVMaximumDischargePowe_L2
maxDischargePower_L3	decimal	0..1	Optional. Maximum discharge power on phase L3 in W, defined by min(EV, EVSE). Value >= 0. Relates to: **ISO 15118-20**: BPT_AC/DC_CPDReqEnergyTransferModeType: EVMaximumDischargePower_L3

Field Name	Field Type	Card.	Description
minChargeCurrent	decimal	0..1	Optional. Minimum charge current in A, defined by max(EV, EVSE) Relates to: **ISO 15118-20**: BPT_DC_CPDReqEnergyTransferModeType: EVMinimumChargeCurrent
maxChargeCurrent	decimal	0..1	Optional. Maximum charge current in A, defined by min(EV, EVSE) Relates to: **ISO 15118-20**: BPT_DC_CPDReqEnergyTransferModeType: EVMaximumChargeCurrent
minDischargeCurrent	decimal	0..1	Optional. Minimum discharge current in A, defined by max(EV, EVSE). Value >= 0. Relates to: **ISO 15118-20**: BPT_DC_CPDReqEnergyTransferModeType: EVMinimumDischargeCurrent
maxDischargeCurrent	decimal	0..1	Optional. Maximum discharge current in A, defined by min(EV, EVSE). Value >= 0. Relates to: **ISO 15118-20**: BPT_DC_CPDReqEnergyTransferModeType: EVMaximumDischargeCurrent
minVoltage	decimal	0..1	Optional. Minimum voltage in V, defined by max(EV, EVSE) Relates to: **ISO 15118-20**: BPT_DC_CPDReqEnergyTransferModeType: EVMinimumVoltage
maxVoltage	decimal	0..1	Optional. Maximum voltage in V, defined by min(EV, EVSE) Relates to: **ISO 15118-20**: BPT_DC_CPDReqEnergyTransferModeType: EVMaximumVoltage
evTargetEnergyRequest	decimal	0..1	Optional. Energy to requested state of charge in Wh Relates to: **ISO 15118-20**: Dynamic/Scheduled_SEReqControlModeType: EVTargetEnergyRequest
evMinEnergyRequest	decimal	0..1	Optional. Energy to minimum allowed state of charge in Wh Relates to: **ISO 15118-20**: Dynamic/Scheduled_SEReqControlModeType: EVMinimumEnergyRequest
evMaxEnergyRequest	decimal	0..1	Optional. Energy to maximum state of charge in Wh Relates to: **ISO 15118-20**: Dynamic/Scheduled_SEReqControlModeType: EVMaximumEnergyRequest
evMinV2XEnergyRequest	decimal	0..1	Optional. Energy (in Wh) to minimum state of charge for cycling (V2X) activity. Positive value means that current state of charge is below V2X range. Relates to: **ISO 15118-20**: Dynamic_SEReqControlModeType: EVMinimumV2XEnergyRequest
evMaxV2XEnergyRequest	decimal	0..1	Optional. Energy (in Wh) to maximum state of charge for cycling (V2X) activity. Negative value indicates that current state of charge is above V2X range. Relates to: **ISO 15118-20**: Dynamic_SEReqControlModeType: EVMaximumV2XEnergyRequest
targetSoC	integer, 0 < = val < = 100	0..1	Optional. Target state of charge at departure as percentage. Relates to: **ISO 15118-20**: BPT_DC_CPDReqEnergyTransferModeType: TargetSOC

2.114. V2XFreqWattPointType

Class

(2.1) A point of a frequency-watt curve.

V2XFreqWattPointType is used by: Common:ChargingSchedulePeriodType

Field Name	Field Type	Card.	Description
frequency	decimal	1..1	Required. Net frequency in Hz.
power	decimal	1..1	Required. Power in W to charge (positive) or discharge (negative) at specified frequency.

2.115. V2XSignalWattPointType

Class

(2.1) A point of a signal-watt curve.

V2XSignalWattPointType is used by: Common:ChargingSchedulePeriodType

Field Name	Field Type	Card.	Description
signal	integer	1..1	Required. Signal value from an AFRRSignalRequest.
power	decimal	1..1	Required. Power in W to charge (positive) or discharge (negative) at specified frequency.

2.116. 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.
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.

2.117. 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.
maxElements	integer, 1 < = val	0..1	Optional. **(2.1)** Maximum number of elements from *valuesList* that are supported as *attributeValue*.

Field Name	Field Type	Card.	Description
valuesList	string[0..1000]	0..1	Optional. Mandatory when *dataType* = OptionList, MemberList or SequenceList. In that case *valuesList* specifies the allowed values for the type. The length of this field can be limited by DeviceDataCtrlr.ConfigurationValueSize. * 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. This is a comma separated list. The Configuration Variable ConfigurationValueSize can be used to limit SetVariableData.attributeValue and VariableCharacteristics.valuesList. The max size of these values will always remain equal.
supportsMonitoring	boolean	1..1	Required. Flag indicating if this variable supports monitoring.

2.118. VariableMonitoringType

Class

A monitoring setting for a variable.

VariableMonitoringType is used by: NotifyMonitoringReportRequest.MonitoringDataType

Field Name	Field Type	Card.	Description
id	integer, 0 < = val	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.
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, 0 < = val	1..1	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. The severity levels have the following meaning: **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.
eventNotificationType	EventNotificationEnumType	1..1	Required. **(2.1)** Type of monitor.

2.119. 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.
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.

2.120. VoltageParamsType

Class

VoltageParamsType is used by: Common:DERCurveType

Field Name	Field Type	Card.	Description
hv10MinMeanValue	decimal	0..1	Optional. EN 50549-1 chapter 4.9.3.4 Voltage threshold for the 10 min time window mean value monitoring. The 10 min mean is recalculated up to every 3 s. If the present voltage is above this threshold for more than the time defined by *hv10MinMeanValue*, the EV must trip. This value is mandatory if *hv10MinMeanTripDelay* is set.
hv10MinMeanTripDelay	decimal	0..1	Optional. Time for which the voltage is allowed to stay above the 10 min mean value. After this time, the EV must trip. This value is mandatory if OverVoltageMeanValue10min is set.
powerDuringCessation	PowerDuringCessationEnumType	0..1	Optional. Parameter is only sent, if the EV has to feed-in power or reactive power during fault-ride through (FRT) as defined by HVMomCess curve and LVMomCess curve.

2.121. VPNType

Class

VPN Configuration settings

VPNType is used by: SetNetworkProfileRequest.NetworkConnectionProfileType

Field Name	Field Type	Card.	Description
server	string[0..2000]	1..1	Required. VPN Server Address
user	string[0..50]	1..1	Required. VPN User
group	string[0..50]	0..1	Optional. VPN group.
password	string[0..64]	1..1	Required. **(2.1)** VPN Password.
key	string[0..255]	1..1	Required. VPN shared secret.
type	VPNEnumType	1..1	Required. Type of VPN

Chapter 3. Enumerations

3.1. APNAuthenticationEnumType

Enumeration

APNAuthenticationEnumType is used by: setNetworkProfile:SetNetworkProfileRequest.APNType

Value	Description
PAP	Use PAP authentication
CHAP	Use CHAP authentication
NONE	Use no 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	<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.

3.5. BatterySwapEventEnumType

Enumeration

(2.1) Battery in/out event at a swap station.

BatterySwapEventEnumType is used by: batterySwap:BatterySwapRequest

Value	Description
BatteryIn	Battery (or set of batteries) is inserted.
BatteryOut	Battery (or set of batteries) is removed.
BatteryOutTimeout	The offered batteries have not been removed within timeout.

3.6. 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.

3.7. 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.

3.8. CertificateActionEnumType

Enumeration

CertificateActionEnumType is used by: get15118EVCertificate:Get15118EVCertificateRequest

Value	Description
Install	Install the provided certificate.
Update	Update the provided certificate.

3.9. CertificateSignedStatusEnumType

Enumeration

CertificateSignedStatusEnumType is used by: certificateSigned:CertificateSignedResponse

Value	Description
Accepted	Signed certificate is valid.
Rejected	Signed certificate is invalid or *requestId* is unknown.

3.10. CertificateSigningUseEnumType

Enumeration

CertificateSigningUseEnumType is used by: signCertificate:SignCertificateRequest , certificateSigned:CertificateSignedRequest

Value	Description
ChargingStationCertificate	Client side certificate used by the Charging Station to connect the the CSMS.
V2GCertificate	Use for certificate for ISO 15118-2 connections. This means that the certificate should be derived from the V2G root.
V2G20Certificate	**(2.1)** Use for certificate for ISO 15118-20 connections. This means that the certificate should be derived from the V2G root.

3.11. CertificateStatusEnumType

Enumeration

OCSP or CRL status of certificate.

CertificateStatusEnumType is used by: Common:CertificateStatusType

Value	Description
Good	Certificate has not been revoked.
Revoked	Certificate has been revoked.
Unknown	Certificate is unknown.
Failed	The request to OCSP responder or CRL distribution point failed.

3.12. CertificateStatusSourceEnumType

Enumeration

Source of certificate status, OCSP or CRL.

CertificateStatusSourceEnumType is used by: Common:CertificateStatusRequestInfoType , Common:CertificateStatusType

Value	Description
CRL	Checked in a certificate revocation list.
OCSP	Checked via OCSP request.

3.13. 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.

3.14. 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 *startSchedule* 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 *startSchedule* is set to a starting point in time.
Relative	Charging schedule periods 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 remain relative to the PowerPathClosed moment. No value for *startSchedule* must be supplied.
Dynamic	**(2.1)** The schedule consists of only one charging schedule period, which is updated dynamically by CSMS.

3.15. ChargingProfilePurposeEnumType

Enumeration

Purpose of the charging profile.

ChargingProfilePurposeEnumType is used by: Common:ChargingProfileType , clearChargingProfile:ClearChargingProfileRequest.ClearChargingProfileType , getChargingProfiles:GetChargingProfilesRequest.ChargingProfileCriterionType

Value	Description
ChargingStationExternalConstraints	Additional constraints from an external source (e.g. an EMS) that will be incorporated into a local power schedule. When applied to evse.Id = 0 it sets a limit to the entire Charging Station. Note: In OCPP 2.0.1 this purpose was only allowed on evse.Id = 0. In OCPP 2.1 it can be set to an individual EVSE.
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.

Value	Description
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.
PriorityCharging	**(2.1)** This profile is used in place of a Tx(Default)Profile, when priority charging is requested, either locally on Charging Station or via a request from CSMS.
LocalGeneration	**(2.1)** This profile adds capacity from local generation. Its capacity is added on top of other charging profiles.

3.16. ChargingProfileStatusEnumType

Enumeration

Status returned in response to SetChargingProfileRequest.

ChargingProfileStatusEnumType is used by: setChargingProfile:SetChargingProfileResponse , pullDynamicScheduleUpdate:PullDynamicScheduleUpdateResponse , updateDynamicSchedule:UpdateDynamicScheduleResponse

Value	Description
Accepted	Request has been accepted and will be executed.
Rejected	Request has not been accepted and will not be executed.

3.17. 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 of 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.

3.18. ChargingStateEnumType

Enumeration

The state of the charging process.

ChargingStateEnumType is used by: Common:TransactionType

Value	Description
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. Authorization is required to proceed to state Charging.
Charging	The contactor of the Connector is closed and energy is flowing to between EVSE and EV.
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 restrictions or local supply power constraints.
Idle	There is no connection between EV and EVSE.

3.19. 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.

3.20. ClearChargingProfileStatusEnumType

Enumeration

Status returned in response to ClearChargingProfileRequest.

ClearChargingProfileStatusEnumType is used by: clearChargingProfile:ClearChargingProfileResponse

Value	Description
Accepted	Request has been accepted.
Unknown	No Charging Profile(s) were found matching the request.

3.21. 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.
Rejected	**(2.1)** Request could not be executed.

3.22. ClearMonitoringStatusEnumType

Enumeration

ClearMonitoringStatusEnumType is used by:
clearVariableMonitoring:ClearVariableMonitoringResponse.ClearMonitoringResultType

Value	Description
Accepted	Monitor successfully cleared.
Rejected	Clearing of monitor rejected.
NotFound	Monitor Id is not found.

3.23. ComponentCriterionEnumType

Enumeration

ComponentCriterionEnumType is used by: getReport:GetReportRequest

Value	Description
Active	Components that are active, i.e. having *Active* = 1
Available	Components that are available, i.e. having *Available* = 1
Enabled	Components that are enabled, i.e. having *Enabled* = 1
Problem	Components that reported a problem, i.e. having *Problem* = 1

3.24. 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).

3.25. ControlModeEnumType

Enumeration

(2.1) ISO 15118-20 service parameter for control mode

ControlModeEnumType is used by: Common:ChargingNeedsType

Value	Description
ScheduledControl	Scheduled control mode, EVSE provides up to three schedules for EV to choose from. EV follows the selected schedule.
DynamicControl	Dynamic control mode, EVSE executes a single schedule by sending setpoints to EV at every interval.

3.26. CostDimensionEnumType

Enumeration

Usage dimension for cost in charging period.

CostDimensionEnumType is used by: Common:CostDimensionType

Value	Description
Energy	Total amount of energy (dis-)charged during this charging period, defined in Wh (kiloWatt-hours). When negative, more energy was feed into the grid then charged into the EV.
MaxCurrent	Sum of the maximum current over all phases, reached during this charging period, defined in A (Ampere).
MinCurrent	Sum of the minimum current over all phases, reached during this charging period, when negative, current has flowed from the EV to the grid. Defined in A (Ampere).
MaxPower	Maximum power reached during this charging period: defined in W (Watt).
MinPower	Minimum power reached during this charging period: defined in W (Watt), when negative, the power has flowed from the EV to the grid.
IdleTIme	Time not charging during this charging period: defined in seconds.

Value	Description
ChargingTime	Time charging during this charging period: defined in seconds.

3.27. CostKindEnumType

Enumeration

CostKindEnumType is used by: Common:CostType

Value	Description
CarbonDioxideEmission	Absolute value. Carbon Dioxide emissions, in grams per kWh
RelativePricePercentage	Relative value. Percentage of renewable generation within total generation.
RenewableGenerationPercentage	Relative value. Price per kWh, as percentage relative to the maximum price stated in any of all tariffs indicated to the EV.

3.28. 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.

3.29. 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.

3.30. DataTransferStatusEnumType

Enumeration

Status in DataTransferResponse.

DataTransferStatusEnumType is used by: dataTransfer:DataTransferResponse

Value	Description
Accepted	Message has been accepted and the contained request is accepted.

Value	Description
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.

3.31. DayOfWeekEnumType

Enumeration

DayOfWeekEnumType is used by: Common:TariffConditionsFixedType , Common:TariffConditionsType

Value	Description
Monday	
Tuesday	
Wednesday	
Thursday	
Friday	
Saturday	
Sunday	

3.32. 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 of its certificate type.
NotFound	Requested resource not found.

3.33. DERControlEnumType

Enumeration

Enumeration of DER controls

DERControlEnumType is used by: Common:DERChargingParametersType , Common:DERCurveGetType , setDERControl:SetDERControlRequest , getDERControl:GetDERControlRequest , clearDERControl:ClearDERControlRequest , notifyDERAlarm:NotifyDERAlarmRequest

Value	Description
EnterService	Enter Service parameters setting
FreqDroop	Frequency droop settings
FreqWatt	Frequency-Watt curve
FixedPFAbsorb	Fixed power factor when absorbing power setting
FixedPFInject	Fixed power factor when injecting power setting
FixedVar	Fixed reactive power setpoint
Gradients	Gradient settings
HFMustTrip	High Frequency Must Trip curve
HFMayTrip	High Frequency May Trip curve (ride-through)
HVMustTrip	High Voltage Must Trip curve
HVMomCess	High Voltage Momentary Cessation curve

Value	Description
HVMayTrip	High Voltage May Trip curve (ride-through)
LimitMaxDischarge	Limit discharge power to percentage of rated discharge power
LFMustTrip	Low Frequency Must Trip curve
LVMustTrip	Low Voltage Must Trip curve
LVMomCess	Low Voltage Momentary Cessation curve
LVMayTrip	Low Voltage May Trip curve (ride-through)
PowerMonitoringMustTrip	Power Monitoring curve according to VDE-AR-N 4105 section 5.5.2
VoltVar	Volt-Var curve
VoltWatt	Volt-Watt curve
WattPF	Watt-PowerFactor curve
WattVar	Watt-Var curve

3.34. DERControlStatusEnumType

Enumeration

DERControlStatusEnumType is used by: setDERControl:SetDERControlResponse , getDERControl:GetDERControlResponse , clearDERControl:ClearDERControlResponse

Value	Description
Accepted	Operation successful
Rejected	Operation failed
NotSupported	Type of DER setting or curve is not supported
NotFound	Type or Id in clear/get request was not found

3.35. DERUnitEnumType

Enumeration

DERUnitEnumType is used by: Common:DERCurveType , Common:FixedVarType

Value	Description
Not_Applicable	No unit applicable (e.g. for ride-through curves)
PctMaxW	Percentage of configured active power
PctMaxVar	Percentage of configured reactive power
PctWAvail	Percentage of available reserve active power
PctVarAvail	Percentage of available reserve reactive power
PctEffectiveV	Percentage of effective voltage

3.36. 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.

Value	Description
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.
LanguageNotSupported	**(2.1)** Message contains one or more languages that are not supported by Charging Station.

3.37. EnergyTransferModeEnumType

Enumeration

Enumeration of energy transfer modes.

EnergyTransferModeEnumType is used by: Common:ChargingNeedsType , authorize:AuthorizeResponse , notifyAllowedEnergyTransfer:NotifyAllowedEnergyTransferRequest

Value	Description
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.
DC	DC charging.
AC_BPT	**(2.1)** AC bidirectional (no DER control), ISO 15118-20
AC_BPT_DER	**(2.1)** AC bidirectional with DER control, ISO 15118-20 (amendment to -20)
AC_DER	**(2.1)** AC charging-only with DER control, ISO 15118-20 (amendment to -20) Note: at time of writing (July 2024) not yet defined for ISO 15118-20.
DC_BPT	**(2.1)** DC bidirectional power transfer, ISO 15118-20
DC_ACDP	**(2.1)** DC via ACDP connector (pantograph), ISO 15118-20
DC_ACDP_BPT	**(2.1)** DC bidirectional via ACDP connector (pantograph), ISO 15118-20
WPT	**(2.1)** Wireless power transfer, ISO 15118-20

3.38. EventNotificationEnumType

Enumeration

Specifies the event notification type of the message.

EventNotificationEnumType is used by: Common:VariableMonitoringType , 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.

3.39. EventTriggerEnumType

Enumeration

EventTriggerEnumType is used by: notifyEvent:NotifyEventRequest.EventDataType

Value	Description
Alerting	Monitored variable has passed a Lower or Upper Threshold. Also used as trigger type for a `HardwiredNotification`.
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

3.40. EvseKindEnumType

Enumeration

EvseKindEnumType is used by: Common:TariffConditionsFixedType , Common:TariffConditionsType

Value	Description
AC	AC current EVSE
DC	DC current EVSE

3.41. 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. If sent before installing the firmware, it indicates the Charging Station is about to reboot to start installing new firmware. If sent after installing the new firmware, it indicates the Charging Station has finished installing, but requires a reboot to activate the new firmware, which will be done automatically when idle. 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.

3.42. 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.

3.43. GenericStatusEnumType

Enumeration

Generic message response status

GenericStatusEnumType is used by: openPeriodicEventStream:OpenPeriodicEventStreamResponse , adjustPeriodicEventStream:AdjustPeriodicEventStreamResponse , getCompositeSchedule:GetCompositeScheduleResponse , notifyEVChargingSchedule:NotifyEVChargingScheduleResponse , signCertificate:SignCertificateResponse , setMonitoringLevel:SetMonitoringLevelResponse , publishFirmware:PublishFirmwareResponse , aFRRSignal:AFRRSignalResponse , requestBatterySwap:RequestBatterySwapResponse , vatNumberValidation:VatNumberValidationResponse

Value	Description
Accepted	Request has been accepted and will be executed.
Rejected	Request has not been accepted and will not be executed.

3.44. GetCertificateIdUseEnumType

Enumeration

GetCertificateIdUseEnumType is used by: Common:CertificateHashDataChainType , getInstalledCertificateIds:GetInstalledCertificateIdsRequest

Value	Description
V2GRootCertificate	Use for certificate of the ISO 15118 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.
OEMRootCertificate	**(2.1)** OEM root certificate for 2-way TLS with EV.

3.45. 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.

3.46. 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.

3.47. 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.

3.48. GetInstalledCertificateStatusEnumType

Enumeration

GetInstalledCertificateStatusEnumType is used by: getInstalledCertificateIds:GetInstalledCertificateIdsResponse

Value	Description
Accepted	Normal successful completion (no errors).
NotFound	Requested resource not found.

3.49. GetVariableStatusEnumType

Enumeration

GetVariableStatusEnumType is used by: getVariables:GetVariablesResponse.GetVariableResultType

Value	Description
Accepted	Variable successfully retrieved.
Rejected	Request is rejected.
UnknownComponent	Component is not known.
UnknownVariable	Variable is not known.
NotSupportedAttributeType	The AttributeType is not supported.

3.50. GridEventFaultEnumType

Enumeration

GridEventFaultEnumType is used by: notifyDERAlarm:NotifyDERAlarmRequest

Value	Description
CurrentImbalance	Current imbalance detected
LocalEmergency	A local emergency detected
LowInputPower	Low input power detected

Value	Description
OverCurrent	Overcurrent detected
OverFrequency	Over frequency detected
OverVoltage	Over voltage detected
PhaseRotation	Phase rotation detected
RemoteEmergency	A remote emergency detected
UnderFrequency	Under frequency detected
UnderVoltage	Under voltage detected
VoltageImbalance	Voltage imbalance detected

3.51. 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.

3.52. 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.

3.53. InstallCertificateUseEnumType

Enumeration

InstallCertificateUseEnumType is used by: installCertificate:InstallCertificateRequest

Value	Description
V2GRootCertificate	Use for certificate of the ISO 15118 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.
ManufacturerRootCertificate	Root certificate for verification of the Manufacturer certificate.
CSMSRootCertificate	Root certificate, used by the CA to sign the CSMS and Charging Station certificate.
OEMRootCertificate	**(2.1)** OEM root certificate for 2-way TLS with EV.

3.54. IslandingDetectionEnumType

Enumeration

Methods of islanding detection

IslandingDetectionEnumType is used by: Common:DERChargingParametersType

Value	Description
NoAntiIslandingSupport	No anti-island detection supported
RoCoF	RoCoF - Rate of Change of Frequency
UVP_OVP	Under/over voltage (UVP/OVP)
UFP_OFP	Under/over frequency (UFP/OFP)
VoltageVectorShift	Voltage Vector Shift
ZeroCrossingDetection	Zero Crossing Detection
OtherPassive	Other passive anti-island detection method supported
ImpedanceMeasurement	Impedance measurement
ImpedanceAtFrequency	Impedance detection at a specific frequency
SlipModeFrequencyShift	Slip-mode frequency shift
SandiaFrequencyShift	Frequency bias/Sandia frequency shift
SandiaVoltageShift	Sandia voltage shift
FrequencyJump	Frequency jump
RCLQFactor	RCL Q factor
OtherActive	Other active anti-island detection method supported

3.55. 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.

3.56. 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.
EV	Measurement taken by EV.
Inlet	For the Charging Station (evseId = 0): measurement at network ("grid") inlet connection of the station. For measurements with evseId > 0, these are measurements taken at the EVSE inlet (This can be useful for a DC charger).
Outlet	Measurement at a Connector. Default value.
Upstream	**(2.1)** Measurement taken from an upstream local grid meter of the premise. This can be useful for charging stations that are connected "behind the meter" of a building, and that are able to read the building energy meter.

3.57. 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.
DataCollectorLog	**(2.1)** The log of sampled measurements from the DataCollector component.

3.58. 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 is 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.

3.59. 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. The measurands with .Setpoint are not measured values, but are a value that should be followed as closely as possible.

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.

Note 4: Measurands that have a direction as part of the name (.Import or .Export) have non-negative values. Measurands with .Setpoint, .Net or .Residual in the name can have negative values.

Note 5: Measurands starting with Display represent the optional ISO 15118-20 DisplayParameters.

MeasurandEnumType is used by: Common:SampledValueType

Value	Description
Current.Export	Instantaneous current flow from EV

Value	Description
Current.Export.Offered	**(2.1)** Maximum current EV is offered to export. Min(EV, EVSE)
Current.Export.Minimum	**(2.1)** Minimum current EV can discharge with. Max(EV, EVSE)
Current.Import	Instantaneous current flow to EV
Current.Import.Offered	**(2.1)** Maximum current offered to EV.
Current.Import.Minimum	**(2.1)** Minimum current EV can be charged with. Max(EV, EVSE).
Current.Offered	Maximum current offered to EV. Synonymous to Current.Import.Offered.
Display.PresentSOC	**(2.1)** Current state of charge of the EV battery.
Display.MinimumSOC	**(2.1)** Minimum State of Charge EV needs after charging of the EV battery the EV to keep throughout the charging session.
Display.TargetSOC	**(2.1)** Target State of Charge of the EV battery EV needs after charging.
Display.MaximumSOC	**(2.1)** The SOC at which the EV will prohibit any further charging.
Display.RemainingTimeToMinimumSOC	**(2.1)** The remaining time it takes to reach the minimum SOC. It is communicated as the offset in seconds from the point in time this value was received from EV.
Display.RemainingTimeToTargetSOC	**(2.1)** The remaining time it takes to reach the TargetSOC. It is communicated as the offset in seconds from the point in time this value was received from EV.
Display.RemainingTimeToMaximumSOC	**(2.1)** The remaining time it takes to reach the maximum SOC. It is communicated as the offset in seconds from the point in time this value was received from EV.
Display.ChargingComplete	**(2.1)** Indication if the charging is complete from EV point of view (value = 1).
Display.BatteryEnergyCapacity	**(2.1)** The calculated amount of electrical Energy in Wh stored in the battery when the displayed SOC equals 100 %.
Display.InletHot	**(2.1)** Inlet temperature too high to accept specific operating condition.
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.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.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.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.Active.Import.CableLoss	**(2.1)** Calculated energy loss after energy meter. Will be reset to 0 at start of transaction. Unit is Wh.
Energy.Active.Import.LocalGeneration.Register	**(2.1)** Cumulative amount of imported energy that was from local generation. Value will be cumulative during a transaction, but is allowed to be reset to 0 at start of a transaction.
Energy.Active.Net	Numerical value read from the “net active electrical energy" (Wh or kWh) register.
Energy.Active.Setpoint.Interval	**(2.1)** Energy during interval when Setpoint would be followed exactly, as calculated by Charging Station. Relevant when Setpoint changes frequently during an interval as result of LocalLoadBalancing or LocalFrequencyControl. Can be negative if energy was exported.
Energy.Apparent.Export	Numerical value read from the "apparent electrical export energy" (VAh or kVAh) register.
Energy.Apparent.Import	Numerical value read from the "apparent electrical import energy" (VAh or kVAh) register.
Energy.Apparent.Net	Numerical value read from the "apparent electrical energy" (VAh or kVAh) 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.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).

Value	Description
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.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.Reactive.Net	Numerical value read from the “net reactive electrical energy" (varh or kvarh) register.
EnergyRequest.Target	**(2.1)** Energy to requested state of charge. (Wh)
EnergyRequest.Minimum	**(2.1)** Energy to minimum allowed state of charge. (Wh)
EnergyRequest.Maximum	**(2.1)** Energy to maximum allowed state of charge. (Wh)
EnergyRequest.Minimum.V2X	**(2.1)** Energy to minimum state of charge for cycling (V2X) activity. Positive value means that current state of charge is below V2X range. (Wh)
EnergyRequest.Maximum.V2X	**(2.1)** Energy to maximum state of charge for cycling (V2X) activity. Negative value means that current state of charge is above V2X range. (Wh)
EnergyRequest.Bulk	**(2.1)** Energy to end of bulk charging. (Wh)
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.Active.Setpoint	**(2.1)** Power setpoint for charging or discharging (negative for discharging), that should be followed as close as possible.
Power.Active.Residual	**(2.1)** Difference between the given charging setpoint and the actual power measured. Can be negative.
Power.Export.Minimum	**(2.1)** Minimum power the EV can be discharged with. Max(EV, EVSE)
Power.Export.Offered	**(2.1)** Power offered to EV for discharging. Min(EV, EVSE)
Power.Factor	Instantaneous power factor of total energy flow
Power.Import.Offered	**(2.1)** Power offered to EV for charging. Min(EV, EVSE)
Power.Import.Minimum	**(2.1)** Minimum power the EV can be charged with. Max(EV, EVSE)
Power.Offered	Maximum power offered to EV. Synonymous to Power.Import.Offered.
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 *location* = Inlet and *evseId* = 0: voltage at charging station grid connection. For_ location_ = Outlet and *evseId* > 0: voltage at EVSE outlet towards the EV.
Voltage.Minimum	**(2.1)** Minimum voltage the EV can be charged or discharged with. Max(EV, EVSE)
Voltage.Maximum	**(2.1)** Maximum voltage the EV can be charged or discharged with. Min(EV, EVSE)

3.60. 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 7-bit 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.
QRCODE	Message content is a text (usually a URL) that Charging Station will display as a QR code on the display. Note: this is not a dynamic QR code and should not be used for payments.

3.61. 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.

3.62. 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 (no transaction active).
Unavailable	Message only to be shown while the Charging Station is in unavailable state.
Suspended	**(2.1)** Message only to be shown when Charging Station (or EV) has suspending the charging during a transaction.
Discharging	**(2.1)** Message only to be shown while the EV is discharging.

3.63. 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.

Value	Description
SignChargingStationCertificate	To trigger a SignCertificate with typeOfCertificate: ChargingStationCertificate.
SignV2GCertificate	To trigger a SignCertificate with typeOfCertificate: V2GCertificate
SignV2G20Certificate	**(2.1)** Same as SignV2GCertificate, but this triggers Charging Station explicitly to only sign V2G certificate for ISO 15118-20.
StatusNotification	To trigger StatusNotification.
TransactionEvent	To trigger TransactionEvent.
SignCombinedCertificate	To trigger a SignCertificate with typeOfCertificate: ChargingStationCertificate AND V2GCertificate
PublishFirmwareStatusNotification	To trigger PublishFirmwareStatusNotification.
CustomTrigger	**(2.1)** To trigger the message referred to in *customTrigger* field.

3.64. MobilityNeedsModeEnumType

Enumeration

(2.1) ISO 15118-20 service parameter for mobility needs mode.

MobilityNeedsModeEnumType is used by: Common:ChargingNeedsType

Value	Description
EVCC	Only EV determines min/target SOC and departure time.
EVCC_SECC	Charging station or CSMS may also update min/target SOC and departure time.

3.65. 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 *value*.
LowerThreshold	Triggers an event notice when the actual value of the Variable drops below *value*.
Delta	Triggers an event notice when the actual value has changed more than plus or minus *value* 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 *value*.
Periodic	Triggers an event notice every *value* seconds interval, starting from the time that this monitor was set.
PeriodicClockAligned	Triggers an event notice every *value* seconds interval, starting from the nearest clock-aligned interval after this monitor was set. For example, a *value* of 900 will trigger event notices at 0, 15, 30 and 45 minutes after the hour, every hour.
TargetDelta	**(2.1)** Triggers an event notice when the actual value differs from the target value more than plus or minus *value* since the time that this monitor was set or since the last time this event notice was sent, whichever was last. Behavior of this type of monitor for a variable that is not numeric, is not defined. Example: when *target* = 100, *value* = 10, then an event is triggered when *actual* < 90 or *actual* > 110.
TargetDeltaRelative	**(2.1)** Triggers an event notice when the actual value differs from the target value more than plus or minus (*value* * target value) since the time that this monitor was set or since the last time this event notice was sent, whichever was last. Behavior of this type of monitor for a variable that is not numeric, is not defined. Example: when *target* = 100, *value* = 0.1, then an event is triggered when *actual* < 90 or *actual* > 110.

3.66. MonitoringBaseEnumType

Enumeration

MonitoringBaseEnumType is used by: setMonitoringBase:SetMonitoringBaseRequest

Value	Description
All	Activate all pre-configured monitors while leaving custom monitors intact, including those that overrule a pre-configured monitor.
FactoryDefault	(Re)activate the default monitors of the charging station and remove all custom monitors.
HardWiredOnly	Removes all custom monitors and disables all pre-configured monitors.

3.67. 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.

3.68. 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.

3.69. NotifyAllowedEnergyTransferStatusEnumType

Enumeration

(2.1) Status result of a NotifyAllowedEnergyTransferRequest

NotifyAllowedEnergyTransferStatusEnumType is used by: notifyAllowedEnergyTransfer:NotifyAllowedEnergyTransferResponse

Value	Description
Accepted	Request has been accepted.
Rejected	Request has been rejected. Should not occur, unless there are some technical problems.

3.70. NotifyEVChargingNeedsStatusEnumType

Enumeration

NotifyEVChargingNeedsStatusEnumType is used by: notifyEVChargingNeeds:NotifyEVChargingNeedsResponse

Value	Description
Accepted	A schedule will be provided momentarily.
Rejected	**(2.1)** Service not available. No charging profile can be provided. For an ISO 15118-20 session this is used to convey that the requested energy transfer type is not possible.
Processing	The CSMS is gathering information to provide a schedule.

Value	Description
NoChargingProfile	**(2.1)** CSMS will not provide a charging profile at this time. CS should not wait for it. For an ISO 15118-20 session this value is used instead of Rejected to differentiate between the situation where no charging profile is available (NoChargingProfile) and requested energy transfer type is not available (Rejected).

3.71. 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
Any	**(2.1)** Use any interface.

3.72. 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
SOAP	Use SOAP for transport of OCPP PDU’s
JSON	Use JSON over WebSockets for transport of OCPP PDU’s

3.73. 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, websocket subprotocol: `ocpp1.6`
OCPP20	No longer in use. The OCPP 2.0 release of OCPP has been withdrawn. The value OCPP20 is treated as OCPP2.0.1.
OCPP201	OCPP version 2.0.1, websocket subprotocol: `ocpp2.0.1`
OCPP21	**(2.1)** OCPP version 2.1, websocket subprotocol: `ocpp2.1`

3.74. 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.

3.75. OperationModeEnumType

Enumeration

(2.1) Operation mode for (bi-)directional charging during a charging schedule period.

OperationModeEnumType is used by: Common:ChargingSchedulePeriodType , Common:TransactionType

Value	Description
Idle	Minimize energy consumption by having the EV either on standby or in sleep.
ChargingOnly	Classic charging or smart charging mode. (default)
CentralSetpoint	Control of setpoint by CSMS or some secondary actor that relays through the CSMS.
ExternalSetpoint	Control of setpoint by an external actor directly on the Charging Station.
ExternalLimits	Control of (dis)charging limits by an external actor on the Charging Station.
CentralFrequency	Frequency support with control by CSMS or some secondary actor that relays through the CSMS.
LocalFrequency	Frequency support with control in the Charging Station.
LocalLoadBalancing	Load-balancing performed by the Charging Station.

3.76. PaymentStatusEnumType

Enumeration

(2.1) Status of the settlement of an ad hoc payment.

PaymentStatusEnumType is used by: notifySettlement:NotifySettlementRequest

Value	Description
Settled	Settled successfully by the PSP.
Canceled	No billable part of the OCPP transaction, cancelation sent to the PSP.
Rejected	Rejected by the PSP.
Failed	Sent after the final attempt that fails due to communication problems.

3.77. 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

Value	Description
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

3.78. PowerDuringCessationEnumType

Enumeration

PowerDuringCessationEnumType is used by: Common:VoltageParamsType

Value	Description
Active	Active power
Reactive	Reactive power

3.79. PreconditioningStatusEnumType

Enumeration

(2.1) Preconditioning status of the battery

PreconditioningStatusEnumType is used by: transactionEvent:TransactionEventRequest

Value	Description
Unknown	No information available on the status of preconditioning
Ready	The battery is preconditioned and ready to react directly on a given setpoint for charging (and discharging when available).
NotReady	Busy with preconditioning the BMS. When done will move to status Ready.
Preconditioning	The battery is not preconditioned and not able to directly react to given setpoint.

3.80. PriorityChargingStatusEnumType

Enumeration

(2.1) Status of a UsePriorityChargingRequest

PriorityChargingStatusEnumType is used by: usePriorityCharging:UsePriorityChargingResponse

Value	Description
Accepted	Request has been accepted.
Rejected	Request has been rejected.
NoProfile	No priority charging profile present.

3.81. 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.
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.

3.82. 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.

3.83. ReasonEnumType

Enumeration

Reason for stopping a transaction.

Each reason has a (Failed) or (Successful) label, that indicates whether this situation refers to a failed or successful charging session according to Charging Station. There may be situations, though, where a transaction is ended normally (e.g. *stoppedReason* = Local), but no energy was delivered because of a failure in EVSE or EV.

ReasonEnumType is used by: Common:TransactionType

Value	Description
DeAuthorized	The transaction was stopped because of the authorization status in the response to a TransactionEventRequest. (Failed)
EmergencyStop	Emergency stop button was used. (Failed)
EnergyLimitReached	**(2.1)** Deprecated, because it stops energy transfer, not the transaction. EV charging session reached a locally enforced maximum energy transfer limit. (Successful)
EVDisconnected	Disconnecting of cable, vehicle moved away from inductive charge unit. (Successful)
GroundFault	A GroundFault has occurred. (Failed)
ImmediateReset	A Reset(Immediate) command was received. (Failed)
MasterPass	The transaction was stopped using a token that belongs to the MasterPassGroupId. (Successful)
Local	Stopped locally on request of the EV Driver at the Charge Point. This is a regular termination of a transaction. Examples: presenting an IdToken tag, pressing a button to stop. (Successful)

Value	Description
LocalOutOfCredit	**(2.1)** Deprecated, because it stops energy transfer, not the transaction. A local credit limit enforced through the Charging Station has been exceeded. (Successful)
Other	Any other reason. (Failed)
OvercurrentFault	A larger than intended electric current has occurred. (Failed)
PowerLoss	Complete loss of power. (Failed)
PowerQuality	Quality of power too low, e.g. voltage too low/high, phase imbalance, etc. (Failed)
Reboot	A locally initiated reset/reboot occurred. (for instance watchdog kicked in). (Failed)
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. (Successful)
SOCLimitReached	**(2.1)** Deprecated, because it stops energy transfer, not the transaction. Electric vehicle has reported reaching a locally enforced maximum battery State of Charge (SOC). (Successful)
StoppedByEV	The transaction was stopped by the EV. (Successful)
TimeLimitReached	**(2.1)** Deprecated, because it stops energy transfer, not the transaction. EV charging session reached a locally enforced time limit. (Successful)
Timeout	EV not connected within timeout. (Failed)
ReqEnergyTransferRejected	**(2.1)** CSMS cannot accept the requested energy transfer type. (Failed)

3.84. RecurrencyKindEnumType

Enumeration

RecurrencyKindEnumType is used by: Common:ChargingProfileType

Value	Description
Daily	The schedule restarts every 24 hours, at the same time as in the *startSchedule*.
Weekly	The schedule restarts every 7 days, at the same time and day-of-the-week as in the *startSchedule*.

3.85. 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.

3.86. 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	Optional. 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: - AvailabilityState. For each EVSE Component: - AvailabilityState. For each Connector Component: - AvailabilityState (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. - Include TechCode and TechInfo where available. All monitored Component.Variables in Critical or Alert state shall also be included. - 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.

3.87. 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.

3.88. ReservationUpdateStatusEnumType

Enumeration

ReservationUpdateStatusEnumType is used by: reservationStatusUpdate:ReservationStatusUpdateRequest

Value	Description
Expired	The reservation is expired.
Removed	The reservation is removed.
NoTransaction	**(2.1)** The reservation was used, but no transaction was started.

3.89. 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.

Value	Description
Rejected	Reservation has not been made. Charging Station is not configured to accept reservations.
Unavailable	Reservation has not been made, because evse, connectors or specified connector are in an unavailable state.

3.90. ResetEnumType

Enumeration

Type of reset requested.

ResetEnumType is used by: reset:ResetRequest

Value	Description
Immediate	Immediate reset of the Charging Station or EVSE.
OnIdle	Delay reset until no more transactions are active.
ImmediateAndResume	**(2.1)** Immediate reset and resume transaction(s) afterwards

3.91. 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.

3.92. 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 then version number of current list.

3.93. SetMonitoringStatusEnumType

Enumeration

SetMonitoringStatusEnumType is used by: setVariableMonitoring:SetVariableMonitoringResponse.SetMonitoringResultType

Value	Description
Accepted	Monitor successfully set.
UnknownComponent	Component is not known.

Value	Description
UnknownVariable	Variable is not known.
UnsupportedMonitorType	Requested monitor type is not supported.
Rejected	Request is rejected.
Duplicate	A monitor already exists for the given type/severity combination.

3.94. 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

3.95. 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.

3.96. TariffChangeStatusEnumType

Enumeration

TariffChangeStatusEnumType is used by: changeTransactionTariff:ChangeTransactionTariffResponse

Value	Description
Accepted	Tariff has been accepted.
Rejected	Tariff has been rejected. More info in *statusInfo*.
TooManyElements	Tariff has too many elements and cannot be processed.
ConditionNotSupported	A condition is not supported, or conditions are not supported at all.
TxNotFound	Transaction does not exist or has already ended
NoCurrencyChange	Cannot change currency during a transaction

3.97. TariffClearStatusEnumType

Enumeration

TariffClearStatusEnumType is used by: Common:ClearTariffsResultType

Value	Description
Accepted	Clearing tariff has been accepted.
Rejected	Clearing tariff has been rejected. More info in *statusInfo*.
NoTariff	No tariff for EVSE of IdToken

3.98. TariffCostEnumType

Enumeration

Type of cost: normal cost calculation, or limited to a min or max value.

TariffCostEnumType is used by: Common:TotalCostType

Value	Description
NormalCost	Cost is result of normal cost calculation.
MinCost	Cost is the minimum cost for this tariff.
MaxCost	Cost is the maximum cost for this tariff.

3.99. TariffGetStatusEnumType

Enumeration

TariffGetStatusEnumType is used by: getTariffs:GetTariffsResponse

Value	Description
Accepted	Tariff has been accepted.
Rejected	Tariff has been rejected. More info in *statusInfo*.
NoTariff	No tariff present on Charging Station or EVSE.

3.100. TariffKindEnumType

Enumeration

TariffKindEnumType is used by: Common:TariffAssignmentType

Value	Description
DefaultTariff	Default tariff
DriverTariff	Driver-specific tariff

3.101. TariffSetStatusEnumType

Enumeration

TariffSetStatusEnumType is used by: setDefaultTariff:SetDefaultTariffResponse

Value	Description
Accepted	Tariff has been accepted.
Rejected	Tariff has been rejected. More info in *statusInfo*.
TooManyElements	Tariff has too many elements and cannot be processed.
ConditionNotSupported	A condition is not supported, or conditions are not supported at all.
DuplicateTariffId	TariffId already exists in Charging Station.

3.102. 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'.

3.103. 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.
NotImplemented	Requested message cannot be sent because it is either not implemented or unknown.

3.104. TriggerReasonEnumType

Enumeration

Reason that triggered a transactionEventRequest.

TriggerReasonEnumType is used by: transactionEvent:TransactionEventRequest

Value	Description
AbnormalCondition	An Abnormal Error or Fault Condition has occurred.
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 *LimitChangeSignificance* by an external actor (e.g. an EMS).
ChargingStateChanged	Charging State changed.
CostLimitReached	**(2.1)** Maximum cost has been reached, as defined by transactionLimit.maxCost.
Deauthorized	The transaction was stopped because of the authorization status in the response to a transactionEventRequest.
EnergyLimitReached	Maximum energy of charging reached as defined by transactionLimit.maxEnergy.
EVCommunicationLost	Communication with EV lost, for example: cable disconnected.
EVConnectTimeout	EV not connected before the connection is timed out.
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.
LimitSet	**(2.1)** Limit of cost/time/energy/SoC for transaction has set or changed.
MeterValueClock	Needed to send a clock aligned meter value
MeterValuePeriodic	Needed to send a periodic meter value
OperationModeChanged	**(2.1)** V2X operation mode has changed (at start of a new charging schedule period).
RemoteStart	A RequestStartTransactionRequest has been sent.

Value	Description
RemoteStop	A RequestStopTransactionRequest has been sent.
ResetCommand	CSMS sent a Reset Charging Station command.
RunningCost	**(2.1)** Trigger used when TransactionEvent is sent (only) to report a running cost update.
SignedDataReceived	Signed data is received from the energy meter.
SoCLimitReached	**(2.1)** State of charge limit has been reached, as defined by transactionLimit.maxSoC.
StopAuthorized	An EV Driver has been authorized to stop charging. For example: By swiping an RFID card.
TariffChanged	**(2.1)** Tariff for transaction has changed.
TariffNotAccepted	**(2.1)** Trigger to notify that EV Driver has not accepted the tariff for transaction. idToken becomes deauthorized.
TimeLimitReached	**(2.1)** Maximum time of charging reached, as defined by transactionLimit.maxTime.
Trigger	Requested by the CSMS via a TriggerMessageRequest.
TxResumed	**(2.1)** Transaction has resumed after reset or power outage.
UnlockCommand	CSMS sent an Unlock Connector command.

3.105. 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.

3.106. 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.

3.107. 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.

3.108. 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.

3.109. 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.

3.110. 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

Chapter 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, like the DataTransfer message and CustomData extensions or CustomTriggers.
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.
V2XChargingCtrlr **(2.1)**	Responsible for configuration relating to V2X charging/discharging. This component exists on the EVSE tier hierarchy.
DCDERCtrlr **(2.1)**	Responsible for configuration relating to DER capabilities of the DC inverter of the EVSE in the Charging Station. The component is located at the EVSE level, since it represents the DER capabilities, also referred to as nameplate information, of the EVSE.
ACDERCtrlr **(2.1)**	Responsible for configuration relating to DER capabilities that the EVSE of the Charging Station can emulate by using ISO 15118-20 ChargeLoop messages to control the inverter in the EV. The component is located at the EVSE level, since it represents the DER capabilities of the EVSE.
BatterySwapCtrlr **(2.1)**	Responsible for configuration relating to Battery swapping.
WebPaymentsCtrlr **(2.1)**	Responsible for configuration of a dynamic QR code for ad hoc payments.

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.

Chapter 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, Authorization Cache, Transaction, Metering, Reservation, Smart Charging, Tariff & Cost, Diagnostics, Display Message, Battery Swapping, Charging Infrastructure, ISO 15118, Bidirectional Charging related, and some additional components / variables that received a dedicated section; ConnectedEV, FrequencySimulator, DataCollector, Device Model hierarchy variables, DCDERCtrlr, ACDERCtrlr.

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.

General requirements

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.

Shortcut notation for variables

A Configuration Variable is a device model variable. Such a variable is identified by:

1. its variable name, and optional variable instance
2. the component it is a part of, and optional component instance
3. the EVSE and optional connector to which component belongs.

Configuration Variables can be referenced by their simplified name, which is the section header in this chapter in which the variable is described, or by their full definition which is described in the following shortcut notation:

- componentName[instance].variableName[instance]

> As an example, the Configuration Variable `TariffAvailable`, which is the "Tariff" instance of the variable "Available" as part of component "TariffCostCtrlr", can also be referred to as:
>
> TariffCostCtrlr.Available[Tariff].

The EVSE and Connector are not part of the shortcut notation. If a Configuration Variable has a value that is specific for an EVSE or Connector, then this explicitly mentioned in the text.

2.1. General

NOTE WebSocket-related variables are described in "OCPP-2.1 Part 4 JSON over WebSockets".

2.1.1. `ActiveNetworkProfile`

Required	yes	
Component	**componentName**	OCPPCommCtrlr

Variable	**variableName**	ActiveNetworkProfile	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	integer
Description	This variable indicates the NetworkConnectionProfile configuration slot the Charging Station currently uses for its connection with the CSMS.		

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 *true*, 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**	s
		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 *NetworkConnectionProfile*.		

2.1.4. `FileTransferProtocols`

Required	yes		
Component	**componentName**	OCPPCommCtrlr	
Variable	**variableName**	FileTransferProtocols	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	MemberList
		valuesList	FTP, FTPS, HTTP, HTTPS, SFTP
Description	List of supported file transfer protocols.		

2.1.5. `HeartbeatInterval`

Required	no		
Component	**componentName**	OCPPCommCtrlr	
Variable	**variableName**	HeartbeatInterval	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**unit**	s
		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
		valuesList	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 valuesList 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. `NetworkConfiguration`

New in OCPP 2.1

NOTE For improved readability the variables below are listed in a concise format.
The component NetworkConfiguration is located at the top tier, i.e. not assigned to a specific EVSE.

For the variables in NetworkConfiguration that are identical to parameters of SetNetworkProfileRequest, the same types and limits are used.
If Charging Station only supports setting the network configuration via the SetNetworkProfileRequest, then all variables are reported as ReadOnly.

Req’d	Component	Instance	Variable	Comments
yes	NetworkConfiguration	[*configurationSlot*]	OcppCsmsUrl	URL of the CSMS (without the Charging Station Identity)
yes	NetworkConfiguration	[*configurationSlot*]	OcppInterface	OptionList of values of OcppInterfaceEnumType.
yes	NetworkConfiguration	[*configurationSlot*]	OcppTransport	OptionList of values of OCPPTransportEnumType. Is always "JSON".
yes	NetworkConfiguration	[*configurationSlot*]	OcppVersion	OptionList of values of OcppVersionEnumType. This field is ignored, but only present here because it is in SetNetworkProfileRequest.
yes	NetworkConfiguration	[*configurationSlot*]	MessageTimeout	Duration in seconds before a message send by the Charging Station via this network connection times out.
yes	NetworkConfiguration	[*configurationSlot*]	SecurityProfile	Security profile level
yes / no	NetworkConfiguration	[*configurationSlot*]	Identity	Required if SecurityCtrlr/Identity is ReadWrite
yes	NetworkConfiguration	[*configurationSlot*]	BasicAuthPassword	Writing to this variable only sets the password for the “instance” *configurationSlot*
no	NetworkConfiguration	[*configurationSlot*]	CsmsRootCertificateHashAlgorithm	References a specific CSMS Root Certificate which has to be contained in the chain. Same variable as CertificateHashDataType.hashAlgorithm.
no	NetworkConfiguration	[*configurationSlot*]	CsmsRootCertificateIssuerKeyHash	References a specific CSMS Root Certificate which has to be contained in the chain. Same variable as CertificateHashDataType.issuerKeyHash.
no	NetworkConfiguration	[*configurationSlot*]	CsmsRootCertificateIssuerNameHash	References a specific CSMS Root Certificate which has to be contained in the chain. Same variable as CertificateHashDataType.issuerNameHash.
no	NetworkConfiguration	[*configurationSlot*]	CsmsRootCertificateSerialNumber	References a specific CSMS Root Certificate which has to be contained in the chain. Same variable as CertificateHashDataType.serialNumber.
yes	NetworkConfiguration	[*configurationSlot*]	VpnEnabled	True: VPN is configured. False: VPN is not configured. If variable is ReadOnly and set to False: VPN is not supported. When False the values of variables "Vpn*" have no meaning.
no	NetworkConfiguration	[*configurationSlot*]	VpnType	Value from VPNEnumType
no	NetworkConfiguration	[*configurationSlot*]	VpnServer	VPN server address
no	NetworkConfiguration	[*configurationSlot*]	VpnUser	VPN user name
no	NetworkConfiguration	[*configurationSlot*]	VpnGroup	VPN group name
no	NetworkConfiguration	[*configurationSlot*]	VpnPassword	VPN password
no	NetworkConfiguration	[*configurationSlot*]	VpnKey	VPN shared secret
yes	NetworkConfiguration	[*configurationSlot*]	ApnEnabled	True: APN is configured. False: APN is not configured. If variable is ReadOnly and set to False: APN is not supported. When False the values of variables "Apn*", SimPin, PreferredNetwork and UseOnlyPreferredNetwork have no meaning.
no	NetworkConfiguration	[*configurationSlot*]	Apn	Access Point Name as URL

Req'd	Component	Instance	Variable	Comments
no	NetworkConfiguration	[*configurationSlot*]	ApnUserName	APN user name
no	NetworkConfiguration	[*configurationSlot*]	ApnPassword	APN password
no	NetworkConfiguration	[*configurationSlot*]	SimPin	SIM card PIN code
no	NetworkConfiguration	[*configurationSlot*]	PreferredNetwork	Preferred network as concatenation of MCC and MNC
no	NetworkConfiguration	[*configurationSlot*]	UseOnlyPreferredNetwork	When true use only the preferred network
no	NetworkConfiguration	[*configurationSlot*]	ApnAuthentication	Value from APNAuthenticationEnumType

Note: The variables CsmsRootCertificateXXX are printed on two lines for readability, but are a single word: e.g. CsmsRootCertificateHashAlgorithm.

2.1.9. `OfflineThreshold`

Required	yes		
Component	**componentName**	OCPPCommCtrlr	
Variable	**variableName**	OfflineThreshold	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**unit**	s
		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.		

2.1.10. `QueueAllMessages`

Required	no		
Component	**componentName**	OCPPCommCtrlr	
Variable	**variableName**	QueueAllMessages	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean
Description	When this variable is set to *true*, the Charging Station will queue all message until they are delivered to the CSMS. When set to *false* the Charging Station will only queue Transaction related messages as required in: E04.FR.01. and other requirements When this variable is the to *true*, 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		

2.1.11. `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.		

2.1.12. `MessageAttemptIntervalTransactionEvent`

Required	yes		
Component	**componentName**	OCPPCommCtrlr	
Variable	**variableName**	MessageAttemptInterval	
	variableInstance	TransactionEvent	
	variableAttributes	**attributeType**	Actual
		mutability	ReadWrite
	variableCharacteristics	**unit**	s
		dataType	integer
Description	How long the Charging Station should wait before resubmitting a TransactionEventRequest message that the CSMS failed to process.		

2.1.13. `UnlockOnEVSideDisconnect`

Required	yes		
Component	**componentName**	OCPPCommCtrlr	
Variable	**variableName**	UnlockOnEVSideDisconnect	
	variableAttributes	**mutability**	ReadWrite/ReadOnly
	variableCharacteristics	**dataType**	boolean
Description	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.		

2.1.14. `WebSocketPingInterval`

This configuration variable is described in "OCPP-2.1 Part 4 JSON over WebSockets".

2.1.15. `ResetRetries`

Required	yes		
Component	**componentName**	OCPPCommCtrlr	
Variable	**variableName**	ResetRetries	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	integer

Description	Number of times to retry a reset of the Charging Station when a reset was unsuccessful.		

2.1.16. `MessageFieldLength`

Required	no		
Component	**componentName**	OCPPCommCtrlr	
Variable	**variableName**	FieldLength	
	variableInstance	<message>.<field>	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	integer
Description	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.		

2.1.17. `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.		

2.1.18. `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.		

2.1.19. `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.		

2.1.20. 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.		

2.1.21. ConfigurationValueSize

Updated in OCPP 2.1

Required	no		
Component	**componentName**	DeviceDataCtrlr	
Variable	**variableName**	ConfigurationValueSize	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	integer
		maxLimit	2500
Description	This Configuration Variable can be used to limit the following fields: SetVariableData.attributeValue and VariableCharacteristics.valuesList. The max size of these values will always remain equal.		

2.1.22. 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.		

2.1.23. `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.		

2.1.24. `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.		

2.1.25. `DateTime`

Required	yes		
Component	**componentName**	ClockCtrlr	
Variable	**variableName**	DateTime	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	DateTime
Description	Contains the current date and time.		

2.1.26. `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.		

2.1.27. `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.		

2.1.28. `TimeOffset`

Required	no		
Component	**componentName**	ClockCtrlr	
Variable	**variableName**	TimeOffset	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	string
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 to show local time and is also used for charging profiles that have *useLocalTime* = true.		

2.1.29. `NextTimeOffsetTransitionDateTime`

Required	no		
Component	**componentName**	ClockCtrlr	
Variable	**variableName**	NextTimeOffsetTransitionDateTime	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	DateTime
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.		

2.1.30. `TimeOffsetNextTransition`

Required	no		
Component	**componentName**	ClockCtrlr	
Variable	**variableName**	TimeOffset	
	variableInstance	NextTransition	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	string
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.		

2.1.31. `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	Via this variable, the Charging Station provides the CSMS with the option to configure a clock source, if more than 1 are implemented. By providing a list of possible sources, the CSO can configure fallback sources. Example: "NTP,Heartbeat" means, use NTP, but when none of the NTP servers responses, use time synchronization via Heartbeat. 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.		

2.1.32. `TimeZone`

Required	no		
Component	**componentName**	ClockCtrlr	
Variable	**variableName**	TimeZone	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	string

Description	Configured current local time zone in the format: "Europe/Oslo", "Asia/Singapore" etc. 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. The time zone is for display purposes to show local time and is also used for charging profiles that have *useLocalTime* = true.

2.1.33. `TimeAdjustmentReportingThreshold`

Required	no		
Component	**componentName**	ClockCtrlr	
Variable	**variableName**	TimeAdjustmentReportingThreshold	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**unit**	s
		dataType	integer
Description	When the clock time is adjusted forwards or backwards for more then TimeAdjustmentReportingThreshold number of seconds, a SecurityEventNotification("SettingSystemTime") is sent by the charging station. A reasonable value is 20 seconds.		

2.1.34. `CustomImplementationEnabled`

Required	no		
Component	**componentName**	CustomizationCtrlr	
Variable	**variableName**	CustomImplementationEnabled	
	variableInstance	<VendorId>	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean
Description	This standard configuration variable is used to enable/disable custom implementations that the Charging Station supports. It is recommended to first check if the custom behavior can be implemented using device model components/variables before resorting to DataTransfer message(s) and/or CustomData fields.		

2.1.35. `CustomTriggers`

New in OCPP 2.1

Required	no		
Component	**componentName**	CustomizationCtrlr	
Variable	**variableName**	CustomTriggers	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	MemberList

Description	This variable defines the names of custom triggers that Charging Station supports in a *customTrigger* field of TriggerMessageRequest. The names are as comma-separated list in the `Actual` *value* of CustomTriggers.

2.1.36. `ExternalConfigChangeDate`

New in OCPP 2.1

Required	no		
Component	**componentName**	OCPPCommCtrl	
Variable	**variableName**	ExternalConfigChangeDate	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	DateTime
Description	Date/time when the configuration was changed externally, i.e. outside of CSMS, for example by a local service action. This can be monitored by CSMS to decide whether to issue a new GetBaseReportRequest to get the updated configuration.		

2.2. Security related

2.2.1. `BasicAuthPassword`

Updated in OCPP 2.1

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	At least 40, at most 64.
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 a maximum as defined by the *maxLimit* of `BasicAuthPassword`, which must be at least 40 characters and at most 64. 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. **(2.1)** This variable has remained for backwards compatibility. As of OCPP 2.1 the variable BasicAuthPassword of component `NetworkConfiguration` must be used. If SecurityCtrlr.BasicAuthPassword is set directly, Charging Station SHALL also set the variable of the same name in all `NetworkConfiguration` component instances to the same value (if valid), including component instances which are contained in the currently configured NetworkConfigurationPriority.		

2.2.2. `Identity`

Updated in OCPP 2.1

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 [ISO15118-2]. **(2.1)** This variable has remained for backwards compatibility. As of OCPP 2.1 the variable Identity of component `NetworkConfiguration` must be used. If SecurityCtrlr.Identity is set directly, Charging Station SHALL also set the variable of the same name in all `NetworkConfiguration` component instances to the same value (if valid), including component instances which are contained in the currently configured NetworkConfigurationPriority.		

2.2.3. OrganizationName

Required	yes		
Component	**componentName**	SecurityCtrlr	
Variable	**variableName**	OrganizationName	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	string
Description	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	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. 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. *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.* *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.*		

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**	s
		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
Description	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.		

2.2.10. `AllowSecurityProfileDowngrade`

Required	no		
Component	**componentName**	SecurityCtrlr	
Variable	**variableName**	AllowSecurityProfileDowngrade	
	variableAttributes	**mutability**	ReadWrite/ReadOnly
	variableCharacteristics	**dataType**	boolean
Description	If this variable is implemented and set to *true*, then the Charging Station allows downgrading the security profile from 3 to 2. For security reasons it is not allowed to revert from profile 2 or profile 3 to security profile 1 using OCPP.		

2.3. Authorization related

2.3.1. `AuthEnabled`

Required	no		
Component	**componentName**	AuthCtrlr	

Variable	**variableName**	Enabled	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean
Description	If set to *false*, 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 *idToken* field of the TransactionEventRequest. If no idToken was provided, then *idToken* in TransactionEventRequest will be left empty and *type* is set to `NoAuthorization`.		

2.3.2. `AdditionalInfoItemsPerMessage`

Required	no		
Component	**componentName**	AuthCtrlr	
Variable	**variableName**	AdditionalInfoItemsPerMessage	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	integer
Description	Maximum number of *additionalInfo* items that can be sent in one message. This configuration variable only has to be implemented when Charging Station is able to **receive** *additionalInfo* fields from CSMS. It is not required when only the Charging Station can send *additionalInfo* fields, because CSMS is expected to be able to handle an arbitrary amount of *additionalInfo* fields from Charging Station.		

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 *true*, 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 *Offline*, 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 *true* this instructs the Charging Station to not issue any AuthorizationRequests, but only use Authorization Cache and Local Authorization List to determine validity of idTokens. Note: The difference between **AuthCtrlr.DisableRemoteAuthorization** and **AuthCacheCtrlr.DisablePostAuthorization** is that the latter only disables re-authorization of tokens that are as not-Accepted in the Authorization Cache or Local Authorization List, whereas **AuthCtrlr.DisableRemoteAuthorization** disables all authorization with CSMS.

2.3.9. `SupportedIdTokentypes`

Required	yes		
Component	**componentName**	AuthCtrlr	
Variable	**variableName**	SupportedIdTokenTypes	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	MemberList
		valuesList	List of IdTokenEnumStringType in Appendix.
Description	The subset of the list of supported IdTokenTypes as defined in Appendix 7. "Standardized values for enumerations as string: IdTokenEnumStringType", that is supported by the Charging Station.		

2.4. Adhoc payment related

New in OCPP 2.1

2.4.1. `PaymentCtrlr`

Below is an overview of variables for component PaymentCtrlr. See use case C18 - Authorization using locally connected payment terminal.

Variable	Type	M/O	R/W	Description
Enabled	boolean	M	RW	Payment terminal support is enabled.
Problem	boolean	M	RO	There’s a problem with the payment terminal.
Authorization and settlement				
AuthorizeDirectPayment	boolean	M	RW	If true, an AuthorizeRequest must be sent to CSMS to approve the direct payment.
AuthorizationAmount	decimal	M	RW	Amount used for the pre-authorization.
IncrementalAuthorizationAmount	decimal	O	RW	If value is 0, then incremental authorization is not allowed. Variable will be absent if Charging Station or payment terminal do not support incremental authorization.
IncrementalAuthorizationThreshold	decimal	O	RW	If costs exceed current authorization amount minus IncrementalAuthorizationThreshold then the authorization needs to be increased. When this variable is absent this threshold is defined by the implementation.

Variable	Type	M/O	R/W	Description
PaymentDetails	MemberList	M	RW	The *valuesList* of PaymentDetails contains the information that the payment terminal is able to provide. The Actual *value* of PaymentDetails determines which of these details shall be provided in the *idToken.additionInfo* field. *valuesList*: "PspRef", "SessionRef", "MerchantRef", "PaymentBrand", "ReadingMethod", "PaymentRecognition", "CardBin", "CardLast4Digits", "CardExpiryDate", "HashedCardNr", "WalletUserId". See also chapter Standardized additionalInfo types in Appendix.
SettlementByCSMS	boolean	M	RW	When true, CSMS will take care of settlement.
Receipts				
ReceiptServerUrl	string	M	RW	URL to the receipt server, where an EV driver can find the receipt afterwards.
ReceiptByCSMS	boolean	M	RW	When true, CSMS will provide a URL to receipt, else it is provided by payment terminal.
Merchant[Id]	string	M	RW	Merchant ID to be added to a PSP/locally generated receipt.
Merchant[TaxId]	string	M	RW	Tax ID of the merchant to be added to a PSP/locally generated receipt.
Merchant[Name]	string	M	RW	Name of the merchant to be added to a PSP/locally generated receipt.
Merchant[Address]	string	M	RW	Address of the merchant to be added to a PSP/locally generated receipt.
Merchant[City]	string	M	RW	City of the merchant to be added to a PSP/locally generated receipt.
Terminal data				
TerminalID	string	M	RO	Terminal ID of the payment terminal.
PaymentServiceProvider	string	M	RO	The payment service provider that the terminal is using. Typically read-only.
VendorName	string	M	RO	Manufacturer of the payment terminal.
Model	string	M	RO	Model of the payment terminal.
SerialNumber	string	M	RO	Payment terminal serial number.
FirmwareVersion	string	M	RO	Payment terminal firmware version.
IMSI	string	M	RO	IMSI of the payment terminals SIM card.
ICCID	string	M	RO	ICCID of the payment terminals SIM card.
Connected	boolean	M	RO	Boolean to indicate whether the payment terminal is connected to its payment service provider.

M = mandatory, O = optional

RO = read-only, RW = read-write
Note: notation "Merchant[xxx]" refers to instance "xxx" of variable "Merchant"

2.4.2. `WebPaymentsCtrlr`

This logical component is used to configure the creation of dynamic QR codes for payment. See use case C25 - Ad hoc payment via static or dynamic QR code

Variable	Type	RW	M/O	Description
URLTemplate	string	RW	M	URL template
URLParameters	MemberList	RO	O	List of supported URL query parameters "maxtime", "maxenergy" and "maxcost". *valuesList*: "maxtime", "maxenergy", "maxcost". When absent, none of these are supported.

Variable	Type	RW	M/O	Description
TOTPVersion	OptionList	RW	M	Version of the TOTP algorithm to use. *valuesList*: "v1" The "TOTP algorithm, version 1" from C25 - Ad hoc payment via a QR code is referenced as "v1". When a Charging Station supports multiple versions, they are listed in *valuesList*.
RoamingEvseId	string	RW	O	Roaming EVSE Id to be used when URL is pointing to an external party.
ValidityTime	integer	RW	M	Validity of a one-time password in seconds. Acceptable range between 6 seconds up to 3600 seconds.
SharedSecret	string	WO	M	<random text> set to a random value on first boot. Must be at least 8 characters long.
Length	integer	RW	M	Length of TOTP, e.g. 8 characters. Must be at least 6 characters.
QRCodeQuality	OptionList	RO/RW	O	The error correction level, defined as: Low, Medium, Quartile, High

M = mandatory, O = optional
RO = read-only, RW = read-write, WO = write-only

2.5. Authorization Cache related

2.5.1. `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 *true*, Authorization Cache is enabled.		

2.5.2. `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 *true*, Authorization Cache is supported, but not necessarily enabled.		

2.5.3. `AuthCacheLifeTime`

Required	no	
Component	**componentName**	AuthCacheCtrlr

Variable	**variableName**	LifeTime	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**unit**	s
		dataType	integer
Description	Indicates how long it takes until a token expires in the authorization cache since it is last used.		

2.5.4. `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.		

2.5.5. `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.		

2.5.6. `AuthCacheDisablePostAuthorize`

Required	no		
Component	**componentName**	AuthCacheCtrlr	
Variable	**variableName**	DisablePostAuthorize	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean
Description	When set to *true* 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.		

2.6. Local Authorization List Management related

2.6.1. `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 *true*, Local Authorization List is enabled.		

2.6.2. `LocalAuthListEntries`

Required	when `LocalAuthListAvailable` is *true*		
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.		

2.6.3. `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 *true*, Local Authorization List is supported.		

2.6.4. `ItemsPerMessageSendLocalList`

Required	when `LocalAuthListAvailable` is *true*		
Component	**componentName**	LocalAuthListCtrlr	
Variable	**variableName**	ItemsPerMessage	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	integer
Description	Maximum number of identifications that can be sent in a single SendLocalListRequest.		

2.6.5. `BytesPerMessageSendLocalList`

Required	when `LocalAuthListAvailable` is *true*		
Component	**componentName**	LocalAuthListCtrlr	
Variable	**variableName**	BytesPerMessage	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	integer
Description	Message Size (in bytes) - puts a constraint on SendLocalListRequest message size.		

2.6.6. `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.		

2.6.7. `LocalAuthListDisablePostAuthorize`

Required	no		
Component	**componentName**	LocalAuthListCtrlr	
Variable	**variableName**	DisablePostAuthorize	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean
Description	When set to *true* 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.		

2.6.8. `LocalAuthListSupportsExpiryDateTime`

Required	no		
Component	**componentName**	LocalAuthListCtrlr	
Variable	**variableName**	SupportsExpiryDateTime	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	boolean
Description	When set to *true* Charging Station will disregard idTokens for authorization as if not present in the Local Authorization List when current date/time is past the value of *cacheExpiryDateTime*. Note, that *cacheExpiryDateTime* does not affect the behavior of SendLocalListRequest or GetLocalListRequest messages.		

2.7. Transaction related

2.7.1. `EVConnectionTimeOut`

Required	yes		
Component	**componentName**	TxCtrlr	
Variable	**variableName**	EVConnectionTimeOut	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**unit**	s
		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.		

2.7.2. `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 *true*, the Charging Station SHALL deauthorize the transaction when the cable is unplugged from the EV.		

2.7.3. `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.		

2.7.4. `TxStartPoint`

Required	yes	
Component	**componentName**	TxCtrlr

Variable	**variableName**	TxStartPoint	
	variableAttributes	**mutability**	ReadOnly or ReadWrite. Choice is up to Charging Station implementation.
	variableCharacteristics	**dataType**	MemberList
		valuesList	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.		

2.7.5. `TxStopPoint`

Required	yes		
Component	**componentName**	TxCtrlr	
Variable	**variableName**	TxStopPoint	
	variableAttributes	**mutability**	ReadOnly or ReadWrite. Choice is up to Charging Station implementation.
	variableCharacteristics	**dataType**	MemberList
		valuesList	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.		

2.7.6. TxStartStopPoint values

2.7.6.1. 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.

Value	Description
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. Note: Since energy needs to start flowing first to cause the transaction to be started, there is a small time gap (order of milliseconds) between the start of energy transfer and start of transaction. Depending on the implementation this may potentially skew the value of the energy meter reading associated with start of the transaction. Use PowerPathClosed as TxStartPoint to avoid this situation.
DataSigned	The moment when the signed meter value is received from the fiscal meter, that is used in the TransactionEventRequest with *context* = `Transaction.Begin` and *triggerReason* = `SignedDataReceived`. 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.

2.7.6.2. 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.
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.
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.

2.7.7. `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.		

2.7.8. `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.		

2.7.9. `TxResumptionTimeout`

New in OCPP 2.1

Required	no		
Component	**componentName**	TxCtrlr	
Variable	**variableName**	ResumptionTimeout	
	variableAttributes	**mutability**	ReadWrite / ReadOnly
	variableCharacteristics	**dataType**	integer
		unit	s
		minLimit	0
Description	This variable defines the maximum number of seconds that a transaction may be interrupted by a power outage and still be resumed afterwards. When absent default to 0.		

2.7.10. `TxAllowEnergyTransferResumption`

New in OCPP 2.1

Required	no		
Component	**componentName**	TxCtrlr	
Variable	**variableName**	AllowEnergyTransferResumption	
	variableAttributes	**mutability**	ReadWrite / ReadOnly
	variableCharacteristics	**dataType**	boolean
Description	This variable defines whether energy transfer is allowed to be resumed when the transaction is resumed after a reset or power outage. Energy transfer will return to the state it had before the reset or power outage. When absent this variable defaults to false.		

2.8. `EnergyTransferResumptionRandomRange`

New in OCPP 2.1

Required	no		
Component	**componentName**	TxCtrlr	
Variable	**variableName**	EnergyTransferResumptionRandomRange	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**unit**	s
		dataType	integer
Description	When the Charging Station gets shut down unexpectedly, e.g. due to a software fault, watchdog event, maintenance mode activation or power loss and the interruption was not longer than `TxResumptionTimeout` seconds and `TxAllowEnergyTransferResumption` = true and before interruption the charging state was `Charging` and the EV and EVSE are not disconnected, then the Charging Station shall delay resuming energy transfer for a random amount of seconds, with a maximum of the value configured at `EnergyTransferResumptionRandomRange`.		

2.8.1. `TxSupportedLimits`

New in OCPP 2.1

Required	no		
Component	**componentName**	TxCtrlr	
Variable	**variableName**	SupportedLimits	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	MemberList
		valuesList	List of "maxCost, maxEnergy, maxTime, maxSoC"
Description	This variable defines which transaction limits in TransactionLimitType are supported by the Charging Station. If the list is empty or this variable does not exist, then Charging Station supports no transaction limits.		

2.9. Metering related

2.9.1. `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.		

2.9.2. `SampledDataAvailable`

Required	no

Component	**componentName**	SampledDataCtrlr	
Variable	**variableName**	Available	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	boolean
Description	If this variable reports a value of *true*, Sampled Data is supported.		

2.9.3. `SampledDataRegisterValuesWithoutPhases`

Required	no		
Component	**componentName**	SampledDataCtrlr	
Variable	**variableName**	RegisterValuesWithoutPhases	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean
Description	If this variable reports a value of *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. If this variable is absent or *false*, then the value for each phase is reported, possibly also with a total value (depending on the meter).		

2.9.4. `SampledDataSignReadings`

Required	no		
Component	**componentName**	SampledDataCtrlr	
Variable	**variableName**	SignReadings	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean
Description	If set to *true*, the Charging Station SHALL include signed meter values in the TransactionEventRequest(Ended) as defined in `SampledDataTxEndedMeasurands`. 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.		

2.9.5. `SampledDataSignStartedReadings`

Required	no		
Component	**componentName**	SampledDataCtrlr	
Variable	**variableName**	SignStartedReadings	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean

Description	If set to *true*, the Charging Station SHALL include signed meter values for *context* = `Transaction.Begin` in the *metervalues* field in the TransactionEventRequest(Started or Updated) as defined in `SampledDataTxStartedMeasurands`. This setting only has an effect if `SampledDataSignReadings` is set to true. When a Charging Station does not support signing meter values, it SHALL NOT report this variable.

2.9.6. `SampledDataSignUpdatedReadings`

Required	no		
Component	**componentName**	SampledDataCtrlr	
Variable	**variableName**	SignUpdatedReadings	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean
Description	If set to *true*, the Charging Station SHALL include signed meter values in the *metervalues* field in the TransactionEventRequest(Updated) as defined in `SampledDataTxUpdatedMeasurands`. This setting only has an effect if `SampledDataSignReadings` is set to true. When a Charging Station does not support signing meter values, it SHALL NOT report this variable.		

2.9.7. `SampledDataTxEndedMeasurands`

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.
		valuesList	values from Measurands
Description	Sampled measurands to be included in the *meterValues* element of TransactionEventRequest (eventType = Ended), every `SampledDataTxEndedInterval` seconds from the start of the transaction until and including the last measurands at the end 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 (eventType = Ended).		

2.9.8. `SampledDataTxEndedInterval`

Required	yes	
Component	**componentName**	SampledDataCtrlr

Variable	**variableName**	TxEndedInterval	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**unit**	s
		dataType	integer
Description	Interval between sampling of metering (or other) data, intended to be transmitted in the TransactionEventRequest (eventType = Ended) message. For transaction data (evseId>0), samples are acquired and transmitted only in the TransactionEventRequest (eventType = Ended) message. A value of "0" (numeric zero), by convention, is to be interpreted to mean that only the values taken at the *start* and *end* of a transaction SHALL be transmitted (no intermediate values). A TxEndedInterval = 0 is recommended, since other values may result in a lot of data to be transmitted in the TransactionEventRequest (eventType = Ended) message.		

2.9.9. `SampledDataTxStartedMeasurands`

Required	yes		
Component	**componentName**	SampledDataCtrlr	
Variable	**variableName**	TxStartedMeasurands	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	MemberList
		valuesList	values from Measurands
		maxLimit	The maximum length of the CSV formatted 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"		

2.9.10. `SampledDataTxUpdatedMeasurands`

Required	yes		
Component	**componentName**	SampledDataCtrlr	
Variable	**variableName**	TxUpdatedMeasurands	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	MemberList
		valuesList	values from Measurands
		maxLimit	The maximum length of the CSV formatted string, to be defined by the implementer.

Description	Sampled measurands to be included in the *meterValues* element of every TransactionEventRequest (eventType = Updated), 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"

2.9.11. `SampledDataTxUpdatedInterval`

Required	yes		
Component	**componentName**	SampledDataCtrlr	
Variable	**variableName**	TxUpdatedInterval	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**unit**	s
		dataType	integer
Description	Interval between sampling of metering (or other) data, intended to be transmitted via TransactionEventRequest (eventType = Updated) messages. For transaction data (evseId>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.		

2.9.12. `SampledDataUpstreamMeasurands`

New in OCPP 2.1

Required	no		
Component	**componentName**	SampledDataCtrlr	
Variable	**variableName**	UpstreamMeasurands	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	MemberList
		valuesList	values from Measurands
		maxLimit	The maximum length of the CSV formatted string, to be defined by the implementer.
		maxElements	Maximum number of values supported in variable.
Description	Sampled measurands to be included in the *meterValues* element of every TransactionEventRequest (eventType = Updated), every `SampledDataUpstreamInterval` seconds from the start of the transaction for location = `Upstream` only. 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"		

2.9.13. `SampledDataUpstreamInterval`

New in OCPP 2.1

Required	no		
Component	**componentName**	SampledDataCtrlr	
Variable	**variableName**	UpstreamInterval	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**unit**	s
		dataType	integer
Description	Interval between sampling of metering (or other) data, intended to be transmitted via TransactionEventRequest (eventType = Updated) messages for location = `Upstream` only. For transaction data (evseId>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.		

2.9.14. `AlignedDataEnabled`

Required	no		
Component	**componentName**	AlignedDataCtrlr	
Variable	**variableName**	Enabled	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean
Description	If this variable reports a value of *true*, Aligned Data is enabled.		

2.9.15. `AlignedDataAvailable`

Required	no		
Component	**componentName**	AlignedDataCtrlr	
Variable	**variableName**	Available	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	boolean
Description	If this variable reports a value of *true*, Aligned Data is supported.		

2.9.16. `AlignedDataMeasurands`

Required	yes	
Component	**componentName**	AlignedDataCtrlr

Variable	**variableName**	Measurands	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	MemberList
		valuesList	values from Measurands
		maxLimit	The maximum length of the CSV formatted string, to be defined by the implementer.
Description	Clock-aligned measurand(s) to be included in MeterValuesRequest or TransactionEventRequest, 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`.		

2.9.17. `AlignedDataInterval`

Required	yes		
Component	**componentName**	AlignedDataCtrlr	
Variable	**variableName**	Interval	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**unit**	s
		dataType	integer
Description	Size (in seconds) of the clock-aligned data interval, intended to be transmitted in the MeterValuesRequest or TransactionEventRequest 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. A value of "0" (numeric zero), by convention, is to be interpreted to mean that no clock-aligned data should be transmitted.		

2.9.18. `AlignedDataSendDuringIdle`

Required	no		
Component	**componentName**	AlignedDataCtrlr	
	evse	*	
Variable	**variableName**	SendDuringIdle	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean
Description	If set to *true*, 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.		

2.9.19. `AlignedDataSignReadings`

Required	no	
Component	**componentName**	AlignedDataCtrlr

Variable	**variableName**	SignReadings	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean
Description	If set to *true*, the Charging Station SHALL include signed meter values in the SampledValueType in the TransactionEventRequest(Ended) for those measurands defined in `AlignedDataTxEndedMeasurands` and optionally in TransactionEventRequest(Updated) as controlled by `AlignedDataSignUpdatedReadings`. When a Charging Station does not support signed meter values, it SHALL NOT report this variable.		

2.9.20. `AlignedDataSignUpdatedReadings`

Required	no		
Component	**componentName**	AlignedDataCtrlr	
Variable	**variableName**	SignUpdatedReadings	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean
Description	If set to *true*, the Charging Station SHALL include signed meter values in the SampledValueType in the TransactionEventRequest(Updated) for those measurands configured in `AlignedDataMeasurands`. This variable only has an effect if `AlignedDataSignReadings` is set to true. When a Charging Station does not support signed meter values, it SHALL NOT report this variable.		

2.9.21. `AlignedDataTxEndedMeasurands`

Required	yes		
Component	**componentName**	AlignedDataCtrlr	
Variable	**variableName**	TxEndedMeasurands	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	MemberList
		valuesList	values from Measurands
		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 *meterValues* element of TransactionEventRequest (eventType = Ended) 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 (eventType = Ended).		

2.9.22. `AlignedDataTxEndedInterval`

Required	yes	
Component	**componentName**	AlignedDataCtrlr

Variable	**variableName**	TxEndedInterval	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**unit**	s
		dataType	integer
Description	Size (in seconds) of the clock-aligned data interval, intended to be transmitted in the TransactionEventRequest (eventType = Ended) 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 intervals are transmitted (if so enabled) at the end of the transaction in 1 TransactionEventRequest (eventType = Ended) message. This is not a recommended practice, since the size of the message can become very large.		

2.9.23. `AlignedDataUpstreamMeasurands`

New in OCPP 2.1

Required	no		
Component	**componentName**	AlignedDataCtrlr	
Variable	**variableName**	UpstreamMeasurands	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	MemberList
		valuesList	values from Measurands
		maxLimit	The maximum length of the CSV formatted string, to be defined by the implementer.
		maxElements	Maximum number of values supported in variable.
Description	Clock-aligned measurand(s) to be included in MeterValuesRequest, every `AlignedDataUpstreamInterval` seconds for location `Upstream` only. 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 `AlignedDataUpstreamMeasurands`.		

2.9.24. `AlignedDataUpstreamInterval`

New in OCPP 2.1

Required	no		
Component	**componentName**	AlignedDataCtrlr	
Variable	**variableName**	UpstreamInterval	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**unit**	s
		dataType	integer

Description	Size (in seconds) of the clock-aligned data interval, intended to be transmitted in the MeterValuesRequest message for location `Upstream` only. 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, 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.

2.9.25. `PublicKeyWithSignedMeterValue`

Required	no		
Component	**componentName**	OCPPCommCtrlr	
Variable	**variableName**	PublicKeyWithSignedMeterValue	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	OptionList
		valuesList	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 instead of omitting the field when the public key is not sent, it is also allowed to send it as an empty string (as was the case in OCPP 2.0.1).		

2.9.26. `FiscalMeterPublicKey`

Required	no		
Component	**componentName**	FiscalMetering	
	evse	<any>	
Variable	**variableName**	PublicKey	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	string
Description	Configuration variable that can be used to retrieve the public key for a meter connected to a specific EVSE. The FiscalMetering component will typically be on the EVSE-tier level. There may be one at the Charging Station tier level for the overall input of the Charging Station, or for a Charging Station with a single EVSE.		

2.10. Reservation related

2.10.1. `ReservationEnabled`

Required	no	
Component	**componentName**	ReservationCtrlr

Variable	**variableName**	Enabled	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean
Description	Whether Reservation is enabled.		

2.10.2. `ReservationAvailable`

Required	no		
Component	**componentName**	ReservationCtrlr	
Variable	**variableName**	Available	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	boolean
Description	Whether Reservation is supported.		

2.10.3. `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 *true*: Charging Station supports Reservation where EVSE id is not specified.		

2.11. Smart Charging related

2.11.1. `SmartChargingEnabled`

Required	no		
Component	**componentName**	SmartChargingCtrlr	
Variable	**variableName**	Enabled	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean
Description	Whether Smart Charging is enabled.		

2.11.2. `SmartChargingAvailable`

Required	no	
Component	**componentName**	SmartChargingCtrlr

Variable	**variableName**	Available	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	boolean
Description	Whether Smart Charging is supported.		

2.11.3. `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.		

2.11.4. `ChargingProfileMaxStackLevel`

Required	yes		
Component	**componentName**	SmartChargingCtrlr	
Variable	**variableName**	ProfileStackLevel	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	integer
Description	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.		

2.11.5. `ChargingScheduleChargingRateUnit`

Required	yes		
Component	**componentName**	SmartChargingCtrlr	
Variable	**variableName**	RateUnit	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	MemberList
		valuesList	A,W
Description	A list of supported quantities for use in a ChargingSchedule.		

2.11.6. `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.		

2.11.7. `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 is able to respond to external control signals that influence charging. If the variable is true, but CSMS has set `ExternalConstraintsProfileDisallowed` = true, then external control signals are only allowed during a charging profile from CSMS with a *chargingProfilePeriod* = `ExternalLimits` or `ExternalSetpoint`.		

2.11.8. `ExternalConstraintsProfileDisallowed`

New in OCPP 2.1

Required	no		
Component	**componentName**	SmartChargingCtrlr	
Variable	**variableName**	ExternalConstraintsProfileDisallowed	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean
Description	Indicates whether a Charging Station allows an external system to submit limits to be represented as a `ChargingStationExternalConstraints` charging profile. When false or absent this configuration variable allows that `ChargingStationExternalConstraints` charging profiles are created. When true this forbids the creation of a `ChargingStationExternalConstraints` charging profile. In that case the only way for an external system to influence charging is when CSMS explicitly allows this during a charging profile with a schedule that contains an `ExternalLimits` or `ExternalSetpoint` period.		

2.11.9. `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.

2.11.10. `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.		

2.11.11. `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.		

2.11.12. `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.		

2.11.13. `ChargingProfilePersistence`

New in OCPP 2.1

Required	no	
Component	**componentName**	SmartChargingCtrlr

Variable	**variableName**	ChargingProfilePersistence	
	variableInstance	"TxProfile", "LocalGeneration", "ChargingStationExternalConstraints"	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	boolean
Description	If an instance of this variable is true, then charging profiles with the *chargingProfilePurpose* mentioned in the **variableInstance** are persistent, i.e. they are stored persistently and will still exist after a reboot. When these variables are not present or false, then these profiles are not stored in persistent memory. Note: ChargingStationMaxProfile, TxDefaultProfile and PriorityCharging are always persistent (as per K01.FR.27)		

2.11.14. `ChargingProfileUpdateRateLimit`

New in OCPP 2.1

Required	no		
Component	**componentName**	SmartChargingCtrlr	
Variable	**variableName**	UpdateRateLimit	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	integer
Description	This configuration key limits how often a persistent charging profile can be updated. It is the minimum duration in seconds between updates of charging profiles of the same *chargingProfilePurpose*. A Charging Station may reject SetChargingProfileRequests that occur too frequently. ChargingStationMaxProfile, TxDefaultProfile and PriorityCharging are always persistent (as per K01.FR.27). Other *chargingProfilePurpose* profiles may be persistent when reported as such in `ChargingProfilePersistence`.		

2.11.15. `SetpointPriority`

New in OCPP 2.1

Required	no		
Component	**componentName**	SmartChargingCtrlr	
Variable	**variableName**	SetpointPriority	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	OptionList
		valuesList	ExternalSystem,CSMS
Description	Defines which *setpoint* shall be used when a `ChargingStationExternalConstraints` profile with a *operationMode* = `ExternalSetpoint` is active, but at the same time a `Tx(Default)Profile` charging profile is also active with a *setpoint*. By default, (when variable is absent) the *setpoint* from `ChargingStationExternalConstraints` takes precedence.		

2.11.16. `MaxExternalConstraintsId`

New in OCPP 2.1

Required	no		
Component	**componentName**	SmartChargingCtrlr	
Variable	**variableName**	MaxExternalConstraintsId	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	integer
Description	Defines the highest value that a charging profile id of a `ChargingStationExternalConstraints` profile in the Charging Station can have. CSMS will use charging profile id's above this value to avoid conflicts with `ChargingStationExternalConstraints` profiles created by the Charging Station.		

2.11.17. `SupportedAdditionalPurposes`

New in OCPP 2.1

Required	no		
Component	**componentName**	SmartChargingCtrlr	
Variable	**variableName**	SupportedAdditionalPurposes	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	MemberList
		valuesList	PriorityCharging, LocalGeneration
Description	This configuration variable lists the additional ChargingProfilePurposeEnumType, that have been introduced in OCPP 2.1, that are supported by the Charging Station. When this variable is absent or an empty list, then the purposes `PriorityCharging` and `LocalGeneration` are not supported.		

2.11.18. `SupportsDynamicProfiles`

New in OCPP 2.1

Required	no		
Component	**componentName**	SmartChargingCtrlr	
Variable	**variableName**	SupportsFeature	
	instanceName	DynamicProfiles	
	variableAttributes	**mutability**	ReadOnly
	hvariableCharacteristics	**dataType**	boolean
Description	When this variable has value True, then the Charging Station supports charging profiles of ChargingProfileKindEnumType `Dynamic`. This implies that the fields *dynUpdateInterval* and *dynUpdateTime* in ChargingProfileType are supported. When this variable is absent or false, dynamic charging profiles are not supported.		

2.11.19. `SupportsMaxOfflineDuration`

New in OCPP 2.1

Required	no		
Component	**componentName**	SmartChargingCtrlr	
Variable	**variableName**	SupportsFeature	
	instanceName	MaxOfflineDuration	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	boolean
Description	When this variable has value True, then the Charging Station supports the fields *maxOfflineDuration* and *invalidAfterOfflineDuration* in ChargingProfileType. When this variable is absent or false then these fields are not supported.		

2.11.20. `SupportsLocalTime`

New in OCPP 2.1

Required	no		
Component	**componentName**	SmartChargingCtrlr	
Variable	**variableName**	SupportsFeature	
	instanceName	UseLocalTime	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	boolean
Description	When this variable has value True, then the Charging Station supports the field *useLocalTime* in ChargingScheduleType. If *useLocalTime* = true in a charging schedule, then the *startSchedule* is treated as unqualified local time of the Charging Station. When this variable is absent or false, the *useLocalTime* is not supported.		

2.11.21. `SupportsRandomizedDelay`

New in OCPP 2.1

Required	no		
Component	**componentName**	SmartChargingCtrlr	
Variable	**variableName**	SupportsFeature	
	instanceName	RandomizedDelay	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	boolean
Description	When this variable has value True, then the Charging Station supports the field *randomizedDelay* in ChargingScheduleType, which will delay the start of each charging schedule period by a random number between 0 and *randomizedDelay*. When this variable is absent or false, then *randomizedDelay* is not supported.		

2.11.22. `SupportsLimitAtSoC`

New in OCPP 2.1

Required	no		
Component	**componentName**	SmartChargingCtrlr	
Variable	**variableName**	SupportsFeature	
	instanceName	LimitAtSoC	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	boolean
Description	When this variable has value True, then the Charging Station supports the field *limitAtSoC* in ChargingScheduleType, which will cap the limit or setpoint in the ChargingSchedulePeriodType by the value of *limitAtSoC.limit*. When this variable is absent or false, then *limitAtSoC* is not supported.		

2.11.23. `SupportsEvseSleep`

New in OCPP 2.1

Required	no		
Component	**componentName**	SmartChargingCtrlr	
Variable	**variableName**	SupportsFeature	
	instanceName	EvseSleep	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	boolean
Description	When reported as true the Charging Station supports the *evseSleep* flag in a ChargingSchedulePeriodType, which requests the EVSE electronics to go to sleep during *operationMode* = 'Idle'. When absent or false this option is not supported.		

2.12. Tariff & Cost related

2.12.1. `TariffAvailable`

Updated in OCPP 2.1

Required	no		
Component	**componentName**	TariffCostCtrlr	
Variable	**variableName**	Available	
	variableInstance	Tariff	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	boolean
Description	If true then the Charging Station supports the TariffType structure, that describes the tariff in a machine-readable format. Note 1: The capability will not be active until `TariffEnabled` is set to true. Note 2: Local cost calculation based on this tariff requires that `CostEnabled` is true.		

2.12.2. `TariffEnabled`

Updated in OCPP 2.1

Required	no		
Component	**componentName**	TariffCostCtrlr	
Variable	**variableName**	Enabled	
	variableInstance	Tariff	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean
Description	If true then the Charging Station has enabled support for the TariffType structure, that describes the tariff in a machine-readable format. Note: Local cost calculation based on this tariff requires that `CostEnabled` is true.		

2.12.3. `TariffFallbackMessage`

Updated in OCPP 2.1

Required for Charging Stations supporting tariff information.

Required	yes		
Component	**componentName**	TariffCostCtrlr	
Variable	**variableName**	TariffFallbackMessage	
	variableInstance	<RFC5646 language code>	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	string
		maxLimit	512
Description	Message (and/or tariff information) to be shown to an EV Driver when there is no driver specific tariff information available.		

2.12.4. `CostAvailable`

Required	no		
Component	**componentName**	TariffCostCtrlr	
Variable	**variableName**	Available	
	variableInstance	Cost	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	boolean
Description	If true, then Charging Station supports local cost calculation. Note: The capability will not be active until `CostEnabled` is set to true.		

2.12.5. `CostEnabled`

Updated in OCPP 2.1

Required	no		
Component	**componentName**	TariffCostCtrlr	
Variable	**variableName**	Enabled	
	variableInstance	Cost	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean
Description	If true, then local cost calculation is enabled on Charging Station. If false, then CSMS is responsible for cost calculation. If `TariffEnabled` is true, then the local cost calculation is performed based on the provided TariffType, else the local cost calculation is based on other non-specified (e.g. proprietary) information.		

2.12.6. `TotalCostFallbackMessage`

Updated in OCPP 2.1

Required for Charging Stations supporting Tariff Information.

Required	yes		
Component	**componentName**	TariffCostCtrlr	
Variable	**variableName**	TotalCostFallbackMessage	
	variableInstance	<RFC5646 language code>	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	string
		maxLimit	512
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.		

2.12.7. `Currency`

Updated in OCPP 2.1

Required	yes, only when `TariffEnabled` = false		
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. Charging Station will omit or ignore this value when `TariffEnabled` = true, because TariffType already contains a currency.		

2.12.8. `OfflineTariffFallbackMessage`

New in OCPP 2.1

Required	no		
Component	**componentName**	TariffCostCtrlr	
Variable	**variableName**	OfflineTariffFallbackMessage	
	variableInstance	<RFC5646 language code>	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	string
		maxLimit	512
Description	Message (and/or tariff information) to be shown to an EV Driver when Charging Station is offline. Language codes shall be specified as RFC5646. The short form is allowed if no regional variant is needed. For example: "en" (English), "es" (Spanish), "fr" (French). As instances aren't dynamic in the device model, the station will need to expose all languages that it supports. A Charging Station that has no language support and does not support multiple languages should report a single variable without an instance.		

2.12.9. `RunningCostEnabled`

New in OCPP 2.1

Required	yes, if `CostAvailable` = true		
Component	**componentName**	TariffCostCtrlr	
Variable	**variableName**	Enabled	
	variableInstance	RunningCost	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean
Description	If true then Charging Station will provide periodic running cost updates in *costDetails* in TransactionEventRequest messages.		

2.12.10. `CostInterval`

New in OCPP 2.1

Required	yes, if `CostAvailable` = true		
Component	**componentName**	TariffCostCtrlr	
Variable	**variableName**	Interval	
	variableInstance	Cost	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**unit**	s
		dataType	integer
		minLimit	>0
Description	Specifies the interval in seconds to use to provide periodic running cost updates during a transaction. The value of *minLimit* given by Charging Station specifies the minimal interval that Charging Station will accept.		

2.12.11. `TariffInterval`

New in OCPP 2.1

Required	yes, if `TariffAvailable` = true and `CostAvailable` = true		
Component	**componentName**	TariffCostCtrlr	
Variable	**variableName**	Interval	
	variableInstance	Tariff	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**unit**	s
		dataType	integer
		minLimit	>=0
Description	Interval specifies the maximum interval in seconds to use when evaluating the conditions regarding current, power and energy of a tariff element during a transaction. The value of *minLimit* given by Charging Station specifies the minimal interval that Charging Station will accept. A value of Interval[Tariff] of 0 means Charging Station will evaluate current, power and energy values using the shortest interval possible for this Charging Station.		

2.12.12. `TariffMaxElements`

New in OCPP 2.1

Required	yes, if `TariffAvailable` = true		
Component	**componentName**	TariffCostCtrlr	
Variable	**variableName**	MaxElements	
	variableInstance	Tariff	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	integer
		minLimit	>=0
Description	Specifies the maximum number of *prices* elements that the Charging Station supports in each TariffEnergyType (*energy*), TariffTimeType (*chargingTime, idleTime*) and TariffFixedType (*fixedFee*) of a TariffType.		

2.12.13. `TariffConditionsSupported`

New in OCPP 2.1

Required	yes, if `TariffAvailable` = true		
Component	**componentName**	TariffCostCtrlr	
Variable	**variableName**	ConditionsSupported	
	variableInstance	Tariff	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	boolean

Description	If set to true the Charging Station supports tariffs with conditions (TariffConditionsType and TariffConditionsFixedType). If set to false, the Charging Station does not support conditions in tariffs.

2.12.14. `HandleFailedTariff`

New in OCPP 2.1

Required	yes, if `TariffAvailable` = true		
Component	**componentName**	TariffCostCtrlr	
Variable	**variableName**	HandleFailedTariff .	
	variableInstance	Tariff	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	OptionList
		memberList	"Deauthorize", "UseDefaultTariff", "CentralCost"
Description	This configuration determines how to act when a driver-specific tariff is received, which cannot be processed. • Deauthorize: deauthorize the idToken, such that no energy will be delivered. • UseDefaultTariff: fallback to the default tariff. • CentralCost: let CSMS perform the cost calculation.		

2.13. Diagnostics related

2.13.1. `MonitoringEnabled`

Required	no		
Component	**componentName**	MonitoringCtrlr	
Variable	**variableName**	Enabled	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean
Description	Whether Monitoring is enabled.		

2.13.2. `MonitoringAvailable`

Required	no		
Component	**componentName**	MonitoringCtrlr	
Variable	**variableName**	Available	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	boolean
Description	Whether Monitoring is supported.		

2.13.3. `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.		

2.13.4. `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.		

2.13.5. `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.		

2.13.6. `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.		

2.13.7. `OfflineMonitoringEventQueuingSeverity`

Required	no		
Component	**componentName**	MonitoringCtrlr	
Variable	**variableName**	OfflineQueuingSeverity	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	integer
Description	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).		

2.13.8. `ActiveMonitoringBase`

Required	no		
Component	**componentName**	MonitoringCtrlr	
Variable	**variableName**	ActiveMonitoringBase	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	OptionList
		valuesList	values from *MonitoringBaseEnumType*
Description	Shows the last set MonitoringBase. Valid values according MonitoringBaseEnumType: `All`, `FactoryDefault`, `HardwiredOnly`.		

2.13.9. `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.		

2.13.10. `MonitoringMaxPeriodicEventStreams`

Required	no		
Component	**componentName**	MonitoringCtrlr	
Variable	**variableName**	MaxPeriodicEventStreams	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	integer

Description	The maximum number of open periodic event streams that Charging Station supports. A periodic event stream is opened by a Charging Station in reaction to a SetVariableMonitoringRequest with a *periodicEventStream* field, or can be opened at its own initiative. This variable is required, when event streams are supported.

2.14. Display Message related

2.14.1. `DisplayMessageEnabled`

Required	no		
Component	**componentName**	DisplayMessageCtrlr	
Variable	**variableName**	Enabled	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean
Description	Whether Display Message is enabled.		

2.14.2. `DisplayMessageAvailable`

Required	no		
Component	**componentName**	DisplayMessageCtrlr	
Variable	**variableName**	Available	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	boolean
Description	Whether Display Message is supported.		

2.14.3. `NumberOfDisplayMessages`

Required	yes		
Component	**componentName**	DisplayMessageCtrlr	
Variable	**variableName**	DisplayMessages	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	integer
		maxLimit	Maximum number of different messages that can configured in this Charging Station simultaneous, via SetDisplayMessageRequest.
Description	Amount of different messages that are currently configured in this Charging Station, via SetDisplayMessageRequest		

2.14.4. `DisplayMessageSupportedFormats`

Required	yes	
Component	**componentName**	DisplayMessageCtrlr

Variable	**variableName**	SupportedFormats	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	MemberList
		valuesList	values from MessageFormatEnumType
Description	List of message formats supported by this Charging Station.		

2.14.5. `DisplayMessageSupportedPriorities`

Required	yes		
Component	**componentName**	DisplayMessageCtrlr	
Variable	**variableName**	SupportedPriorities	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	MemberList
		valuesList	values from MessagePriorityEnumType
Description	List of the priorities supported by this Charging Station.		

2.14.6. `DisplayMessageSupportedStates`

New in OCPP 2.1

Required	yes		
Component	**componentName**	DisplayMessageCtrlr	
Variable	**variableName**	SupportedStates	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	MemberList
		valuesList	values from MessageStateEnumType
Description	List of the states during which to display a message, supported by this Charging Station.		

2.14.7. `DisplayMessageLanguage`

New in OCPP 2.1

Required	yes		
Component	**componentName**	DisplayMessageCtrlr	
Variable	**variableName**	Language	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	OptionList
		valuesList	values from RFC 5646
Description	Default language code, per RFC 5646, of this Charging Station. Languages that are supported are listed in the *valuesList*.		

2.15. Charging Infrastructure related

2.15.1. `Available`

Required	yes		
Components	**componentName**	ChargingStation	
		EVSE	
		Connector	
	evse	* (*for EVSE and Connector*)	
Variable	**variableName**	Available	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	boolean
Description	When *true* 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	If any other variables are reported for a Component, then reporting *Available* 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'. 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.		

2.15.2. `AvailabilityState`

Required	yes		
Components	**componentName**	ChargingStation	
		EVSE	
		Connector	
	evse	* (*for EVSE and Connector*)	
Variable	**variableName**	AvailabilityState	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	optionList
		valuesList	Available, Occupied, Reserved, Unavailable, Faulted
Description	This variable reports current availability state for the ChargingStation, EVSE and Connector. When this variable reports the Connector AvailabilityState, it replicates the connectorStatus values as would be reported by the StatusNotification messages. An EVSE or Connector component is addressed on its own tier. So, EVSE #1 is addressed as component EVSE on tier *evse.id* = 1, and EVSE #1, Connector #1 is addressed as component Connector on tier *evse.id* = 1, *evse.connectorId* = 1.		

2.15.3. `AllowReset`

Required	no	
Component	**componentName**	EVSE
	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.		

2.15.4. `ConnectorType`

Required	yes		
Component	**componentName**	Connector	
	evse	*	
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`.		

2.15.5. `PhaseRotation`

Required	no		
Component	**componentName**	*	
	evse	*	
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)

2.15.6. `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.		

2.15.7. `DCInputPhaseControl`

New in OCPP 2.1

Required	no		
Component	**componentName**	EVSE	
	evse	*	
Variable	**variableName**	DCInputPhaseControl	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	boolean

Description	When DCInputPhaseControl is true, then the values of *numberPhases* and *PhaseToUse* in a ChargingSchedulePeriodType will select the input phases from the grid to be used by the DC EVSE.

2.15.8. `Power`

Required	yes (*maxLimit only*)		
Component	**componentName**	EVSE	
	evse	*	
Variable	**variableName**	Power	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	decimal
		maxLimit	decimal
Description	The variableCharacteristic *maxLimit*, that holds the maximum power that this EVSE can provide, is required. The *Actual* value of the instantaneous (real) power is desired, but not required.		

2.15.9. `DischargePower`

New in OCPP 2.1

Required	no		
Component	**componentName**	EVSE	
	evse	*	
Variable	**variableName**	DischargePower	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	decimal
		maxLimit	decimal
		maxSet	decimal
Description	This variable is required when V2X is supported. The variableCharacteristic *maxLimit* holds the maximum rated discharge power that this EVSE can provide. The variableCharacteristic *maxSet* holds the maximum configured discharge power that this EVSE can provide. The *Actual* value of the instantaneous (real) discharge power is recommended to be supported, but not required. Discharge power is represented by a positive value.		

2.15.10. `EvseChargingState`

New in OCPP 2.1

Required	no		
Component	**componentName**	EVSE	
	evse	*	
Variable	**variableName**	ChargingState	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	OptionList
		valuesList	"EVConnected", "Charging", "SuspendedEV", "SuspendedEVSE", "Idle"

Description	This variable reports the current transaction charging state for an EVSE.

2.15.11. `EvseActiveTransactionId`

New in OCPP 2.1

Required	no		
Component	**componentName**	EVSE	
	evse	*	
Variable	**variableName**	ActiveTransactionId	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	string
Description	This variable contains the transaction ID of the running transaction at the EVSE. If no transaction is active, it is an empty string.		

2.15.12. `ChargingStationActiveTransactionId`

New in OCPP 2.1

Required	no		
Component	**componentName**	ChargingStation	
Variable	**variableName**	ActiveTransactionId	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	string
Description	This variable for the ChargingStation component contains a comma-separated list of transaction IDs actively running on the Charging Station. This list may contain transaction IDs that are not part of `EvseActiveTransactionId` when the transaction has not yet been assigned to an EVSE.		

2.15.13. 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 connectorId**	**instance**	**name**	**instance**	**type**	**value**	**dataType**	**maxLimit**	**supports Monitoring**
ChargingStation				Available		Actual	true	boolean		false
ChargingStation				AvailabilityState		Actual	Available	boolean		false
ChargingStation				SupplyPhases		Actual	3	integer		false
ChargingStation				ACCurrent	"L1"	Actual	45.0	decimal		true
ChargingStation				ACCurrent	"L2"	Actual	44.9	decimal		true
ChargingStation				ACCurrent	"L3"	Actual	44.9	decimal		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

Component				**Variable**		**VariableAttribute**		**VariableCharacteristics**		
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

2.16. ISO 15118 related

2.16.1. `CentralContractValidationAllowed`

Required	no		
Component	**componentName**	ISO15118Ctrlr	
Variable	**variableName**	CentralContractValidationAllowed	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean
Description	If this variable exists and has the value *true*, then Charging Station can provide a contract certificate that it cannot validate, to the CSMS for validation as part of the AuthorizeRequest.		

2.16.2. `ContractValidationOffline`

Required	yes		
Component	**componentName**	ISO15118Ctrlr	
Variable	**variableName**	ContractValidationOffline	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean
Description	If this variable is *true*, then Charging Station will try to validate a contract certificate when it is offline.		

2.16.3. `ISO15118PnCEnabled`

Required	no	
Component	**componentName**	ISO15118Ctrlr

Variable	**variableName**	PnCEnabled	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean
Description	If this variable is *true*, then ISO 15118 plug and charge as described by use case C07 - Authorization using Contract Certificates is enabled. If this variable is *false*, then ISO 15118 plug and charge as described by use case C07 - Authorization using Contract Certificates is disabled.		

2.16.4. ISO15118V2GCertificateInstallationEnabled

Required	no		
Component	**componentName**	ISO15118Ctrlr	
Variable	**variableName**	V2GCertificateInstallationEnabled	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean
Description	If this variable is *true*, 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 *false*, 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.		

2.16.5. ISO15118ContractCertificateInstallationEnabled

Required	no		
Component	**componentName**	ISO15118Ctrlr	
Variable	**variableName**	ContractCertificateInstallationEnabled	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean
Description	If this variable is *true*, 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 *false*, then ISO 15118 contract certificate installation/update as described by use case M01 - Certificate installation EV and M02 - Certificate Update EV is disabled.		

2.16.6. ISO15118RequestMeteringReceipt

Required	no		
Component	**componentName**	ISO15118Ctrlr	
Variable	**variableName**	RequestMeteringReceipt	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean

Description	If this variable is *true*, then Charging Station shall request a metering receipt from EV before sending a fiscal meter value to CSMS.

2.16.7. 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. Maximum length of a **commonName** is 64 characters. Example: "DE-ICE-S-0003C4D5578786756453309675436-2"		

2.16.8. 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"		

2.16.9. ISO15118OrganizationName

Required	no		
Component	**componentName**	ISO15118Ctrlr	
	evse	* (optional)	
Variable	**variableName**	OrganizationName	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	string
Description	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.		

2.16.10. ISO15118EvseId

Required	no

Component	**componentName**	EVSE	
	evse	*	
Variable	**variableName**	ISO15118EvseId	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	string
Description	The name of the EVSE in the string format as required by ISO 15118 and IEC 63119-2. Example: "DE*ICE*E*1234567890*1"		

2.16.11. ISO15118NotificationMaxDelay

New in OCPP 2.1

Required	no		
Component	**componentName**	ISO15118Ctrlr	
	evse	*	
Variable	**variableName**	NotificationMaxDelay	
	variableAttributes	**mutability**	ReadOnly, ReadWrite
	variableCharacteristics	**dataType**	integer
Description	The SECC (EVSE) uses the NotificationMaxDelay element in the EVSEStatus to indicate the time in seconds until it expects the EVCC (EV) to react on the action request indicated in EVSENotification.		

2.16.12. ISO15118ServiceRenegotiationSupport

New in OCPP 2.1

Required	no		
Component	**componentName**	ISO15118Ctrlr	
	evse	*	
Variable	**variableName**	ServiceRenegotiationSupport	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	boolean
Description	If set to 'True' the SECC (EVSE) is capable of ServiceRenegotiation. When absent defaults to 'False'.		

2.16.13. ISO15118SupportedProviders

New in OCPP 2.1

Required	no	
Component	**componentName**	ISO15118Ctrlr
	evse	*

Variable	**variableName**	SupportedProviders	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	string
Description	A comma-separated list of all providers (eMSPs) that are supported on this Charging Station. The providers are listed using country and provider ID from the EMAID, as defined in ISO 15118-20. When receiving an ISO 15118-20 AuthorizationSetupReq from EV, charging station may use this string to return a list of supported providers.		

2.16.14. `ISO15118MaxPriceElements`

New in OCPP 2.1

Required	no		
Component	**componentName**	ISO15118Ctrlr	
	evse	*	
Variable	**variableName**	MaxPriceElements	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	integer
		maxLimit	1024
		minLimit	12
Description	The maximum number of *priceRuleStacks* and *priceLevelScheduleEntries* that Charging Station is able to accept in a ChargingScheduleType. ISO 15118 requires support for a minimum of 12 entries.		

2.16.15. `ISO15118ProtocolSupported`

New in OCPP 2.1

Required	no		
Component	**componentName**	ISO15118Ctrlr	
	evse	*	
Variable	**variableName**	ProtocolSupported	
	variableInstance	<for number of protocols supported>	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	string
Description	A string with the following comma-separated items: “<uri>,<major>,<minor>”. <uri> is in the format as used in the SupportedAppProtocolReq message from ISO 15118-2 and ISO 15118-20. This variable has at most 20 instances; one for each supported protocol version. Instances are numbered from 1 to 20 (at most) as a string, i.e. "1", "2", etc, but unlike the variable ProcotolSupportedByEV from ConnectedEV for ISO 15118 this does not specify a priority, since priority is given by EV. Example: - ISO15118Ctrlr.ProtocolSupported["1"] = "urn:iso:15118:2:2013:MsgDef,2,0" - ISO15118Ctrlr.ProtocolSupported["2"] = "urn:iso:std:iso:15118:-20:DC,1,0"		

2.17. Connected EV related

New in OCPP 2.1

This section is new, but the ConnectedEV component already existed in OCPP 2.0.1. The full list of variables of the ConnectedEV component has been moved from part 2 - Appendices, to part 2 Specification - Referenced Components and Variables.

2.17.1. ConnectedEV for ISO 15118

Updated in OCPP 2.1

Required	When supporting V2X	
Component	componentName	ConnectedEV
	evse	* (EVSE to which EV is connected)
Description		
ConnectedEV is a component that represents a connected vehicle for which data is received via an ISO 15118 or CHAdeMO interface. The information that is received, is represented as variables of ConnectedEV.		

Required variables

Variable	Unit	ISO 15118-2 value	ISO 15118-20 value
Available	boolean	When true this means an EV is connected. When false, all other fields are meaningless and should be empty strings.	
Protocol and static vehicle information			
VehicleId	string	EVCCID (from SessionSetupReq) *Six bytes, represented as hexbinary encoded string, e.g. "010203040A0B", containing the EVCC MAC address.*	EVCCID (from SesstionSetupReq) *Max 255 bytes with the first 3 bytes containing the WMI as defined in ISO 3780:2009.*
VehicleCertificate["Leaf"]	string	n/a	The PEM encoded X.509 Leaf certificate of the vehicle certificate chain
VehicleCertificate["SubCA1"]	string	n/a	The PEM encoded X.509 intermediate SubCA1 certificate when present in the vehicle certificate chain
VehicleCertificate["SubCA2"]	string	n/a	The PEM encoded X.509 intermediate SubCA2 certificate, when present in the vehicle certificate chain
VehicleCertificate["Root"]	string	n/a	Optional. The PEM encoded X.509 Root certificate of the vehicle certificate chain
ProtocolAgreed	string	A string with the following comma-separated items: “<uri>,<major>,<minor>”. This is the protocol uri and version information agreed upon between EV and EVSE in the supportedAppProtocolReq handshake from ISO 15118. Example for ISO 15118-2: "urn:iso:15118:2:2013:MsgDef,1,0" (1,0 = version 1.0) For DIN SPEC 70121: "urn:din:70121:2012:MsgDef"	Same definition. Example for ISO 15118-20: "urn:iso:std:iso:15118:-20:DC,1,0" (or AC, ACDP, WPT, etc.)

Variable	**Unit**	**ISO 15118-2 value**	**ISO 15118-20 value**
ProtocolSupportedByEV["1"] to ProtocolSupportedByEV["20"]	multi-instance string	A string with the following comma-separated items: “<uri>,<major>,<minor>”. This is information from the SupportedAppProtocolReq message from ISO 15118 This variable has 20 instances; one for each priority number. Priority is a number from 1 to 20 as a string, i.e. "1", "2", etc. Example: - ConnectedEV.ProtocolSupportedByEV["1"] = "urn:iso:15118:2:2013:MsgDef,2,0" - ConnectedEV.ProtocolSupportedByEV["2"] = "urn:iso:15118:2:2010:MsgDef,1,0"	Same definition. Example for ISO 15118-20: "urn:iso:std:iso:15118:-20:DC,1,0" (or AC, ACDP, WPT, etc.)

Optional variables

Variable	**Unit**	**ISO 15118-2 value**	**ISO 15118-20 value**
Voltage and current values: (*from ChargeParameterDiscoveryReq unless stated otherwise*)			
ACCurrent.minSet	A	EVMinCurrent	-
ACCurrent.maxSet	A	EVMaxCurrent	-
ACVoltage.maxSet	V	EVMaxVoltage	-
DCCurrent.minSet	A	-	EVMinimumChargeCurrent
DCCurrent.maxSet	A	EVMaximumCurrentLimit	EVMaximumChargeCurrent
DCCurrent.target	A	EVTargetCurrent (*CurrentDemandReq*)	EVTargetCurrent (*DC_ChargeLoopReq*)
DCVoltage.minSet	V	-	EVMinimumVoltage
DCVoltage.maxSet	V	EVMaximumVoltageLimit	EVMaximumVoltage
DCVoltage.target	V	EVTargetVoltage (*CurrentDemandReq*)	EVTargetVoltage (*DC_ChargeLoopReq*)
Power, energy and time values:			
Power.maxSet	W	EVMaximumPowerLimit (DC)	EVMaximumChargePower (AC/DC) *All phases combined.*
DischargePower.maxSet	W	-	EVMaximumDischargePower (AC/DC) *Rated maximum, all phases combined* (Actual discharging power is reported via EVSE)
DischargePower.actual	W	-	EVMaximumDischargePower (AC/DC) (*BPT ChargeLoopReq*) *Actual maximum, all phases combined*
EnergyImport.maxSet	Wh	EVEnergyCapacity (DC)	EVMaximumEnergyRequest (*ScheduleExchangeReq, AC/DC_ChargeLoopReq*)
EnergyImport.minSet	Wh	-	EVMinimumEnergyRequest (*ScheduleExchangeReq, AC/DC_ChargeLoopReq*)
EnergyImport.target	Wh	EVEnergyRequest (DC) EAmount (AC)	EVTargetEnergyRequest (*ScheduleExchangeReq, AC/DC_ChargeLoopReq*)
BatteryCapacity	Wh	EVEnergyCapacity (DC)	BatteryCapacity (*AC/DC_ChargeLoopReq.DisplayParameters*)
DepartureTime	dateTime	DepartureTime *Provided as seconds since message receipt. Converted to absolute time.*	DepartureTime (*ScheduleExchangeReq, AC/DC_ChargeLoopReq*) *Provided as seconds since message receipt. Converted to absolute time.*
RemainingTimeBulk	s	RemainingTimeToBulkSoC (*CurrentDemandReq*)	-

Variable	Unit	ISO 15118-2 value	ISO 15118-20 value
RemainingTimeFull.maxSet	s	-	-
RemainingTimeFull.actual	s	RemainingTimeToFullSoc (*CurrentDemandReq*)	RemainingTimeToMaximumSOC (*AC/DC_ChargeLoopReq.DisplayParameters*)
StateOfChargeBulk	%	BulkSoC	-
StateOfCharge.maxSet	%	FullSoC	MaximumSOC (*AC/DC_ChargeLoopReq.DisplayParameters*)
StateOfCharge.actual	%	DC_EVStatus.EVRESSSOC	PresentSOC (*AC/DC_ChargeLoopReq.DisplayParameters*)
ChargingCompleteBulk	boolean	BulkChargingComplete	-
ChargingCompleteFull	boolean	ChargingComplete	ChargingComplete (*AC/DC_ChargeLoopReq.DisplayParameters*)
Error status values:			
ChargingState *with a memberlist consisting of the following values:*			
* *BatteryOvervoltage*		-	<EVTerminationCode>
* *BatteryUndervoltage*		-	…
* *ChargingCurrentDeviation*		FAILED_ChargingCurrentDifferential	…
* *BatteryTemperature*		FAILED_RESSTemperatureInhibit	…
* *VoltageDeviation*		FAILED_ChargingVoltageOutOfRange	…
* *ChargingSystemError*		FAILED_EVRESSMalfunction	…
* *VehicleShiftPosition*		FAILED_EVShiftPosition	…
* *VehicleChargingEnabled*		-	…
* *ChargingSystemIncompatibility*		FAILED_ChargingSystemIncompatibility	…
* *ChargerConnectorLockFault*		FAILED_ChargerConnectorLockFault	…
NOTE 1: ChargingState variable reports an error status and is not related to the *chargingState* in a TransactionEventRequest. **NOTE 2:** ISO 15118-20 uses the EVTerminationCode from the SessionStopReq message to signal what has caused the session to end. Future revisions of ISO 15118-20 will introduce concrete EVTerminationCode values (e.g. urn:iso:std:iso:15118:-20:EVTerminationCode:ChargerConnectorLockFault).			

2.17.2. ConnectedEV for CHAdeMO

Required variables

Variable	Unit	CHAdeMO value
Available	boolean	When true this means an EV is connected. When false, all other fields are meaningless and should be empty strings.
Vehicle ID:		
VehicleId	string	Vehicle ID (H'710 + H'711 + H'712) *Three times 8 bytes, represented as hexbinary encoded string, e.g. "**010203040A0B0C0D**111213141A1B1C1D**212223242A2B2C2D**". A concatenation of **H'710** + H'711 + **H'712**.*
ProtocolAgreed	string	Lowest of Chademo protocol number from EV (H'102.0) and charger (H'109.0) Example for CHAdeMO 2.0.1: "CHAdeMO,3" (3 = version reported by CHAdeMO for v2.0.1)

Variable	Unit	CHAdeMO value
ProtocolSupportedByEV["1"] to ProtocolSupportedByEV["20"]	multi-instance string	Chademo protocol number (H'102.0) supported by EV. For Chademo only the first instance ProtocolSupportedByEV["1"] is used. The other instances have an empty string as value. Reported as a string with the following comma-separated items: "CHAdeMO,<version>". Example for CHAdeMO 2.0.1: - ConnectedEV.ProtocolSupportedByEV["1"] = "CHAdeMO,3"

Optional variables

Variable	Unit	CHAdeMO value
Voltage and current values:		
ACCurrent.minSet	A	-
ACCurrent.maxSet	A	-
ACVoltage.maxSet	V	-
DCCurrent.minSet	A	Minimum charge current (H'100.0)
DCCurrent.maxSet	A	-
DCCurrent.target	A	Charging current request (H'102.3) *If HighCurrentControl is true, use the value from* Charging current request (extended) (H'110.1,2).
DCVoltage.minSet	V	Minimum battery voltage (H'100.2,3)
DCVoltage.maxSet	V	Maximum battery voltage (H'100.4,5)
DCVoltage.target	V	Target battery voltage (H'102.1,2)
Power, energy and time values:		
Power.maxSet	W	-
EnergyImport.maxSet	Wh	Total capacity of traction battery * 100 (H'101.5,6)
EnergyImport.target	Wh	-
DepartureTime	dateTime	-
RemainingTimeBulk	s	-
RemainingTimeFull.maxSet	s	Maximum charging time * 60 (H'101.2)
RemainingTimeFull.actual	s	Estimated charging time * 60 (H'101.3)
StateOfChargeBulk	%	-
StateOfCharge.maxSet	%	Charged rate reference constant (H'100.6)
StateOfCharge.actual	%	State of charge (H'102.6)
ChargingCompleteBulk	boolean	-
ChargingCompleteFull	boolean	-
Error status values:		
ChargingState *with a memberlist consisting of the following values:*		
* *BatteryOvervoltage*		Battery overvoltage (H'102.4.0)
* *BatteryUndervoltage*		Battery undervoltage (H'102.4.1)
* *ChargingCurrentDeviation*		Battery current deviation (H'102.4.2)
* *BatteryTemperature*		High battery temperature (H'102.4.3)
* *VoltageDeviation*		Battery voltage deviation (H'102.4.4)
* *ChargingSystemError*		Charging system error (H'102.5.2)
* *VehicleShiftPosition*		Vehicle shift position (H'102.5.1)
* *VehicleChargingEnabled*		Vehicle charging enabled (H'102.5.0)
* *ChargingSystemIncompatibility*		-
* *ChargerConnectorLockFault*		-
NOTE: ChargingState variable reports an error status and is not related to the *chargingState* in a TransactionEventRequest.		

2.17.3. ConnectedEV for WPT

New in OCPP 2.1

The following parameters coming from the WPT_ChargeLoopReq messages can be represented in the ConnectedEV component. This information might be useful for troubleshooting, but is not required for normal operation.

Optional variables

Variable	Unit	ISO 15118-20 value
WPT_EVPCPowerRequest	decimal	Power the EVPC would like to have as output in Watt.
WPT_EVPCPowerOutput	decimal	Power measured at the output of the EVPC electronics in Watt.
WPT_EVPCChargeDiagnostics	OptionList	Values are: "EVPCNoIssue", "EVPCTempOverheatDetected", "EVPCPowerTransferAnomalyDetected", "EVPCAnomalyDetected"
WPT_EVPCOperatingFrequency	decimal	EVPC measured MF-WPT operating frequency.
WPT_EVPCCoilCurrentRequest	decimal	EVPC wants the primary device to set a specific (preferred) coil current value.
WPT_EVPCCoilCurrentInformation	decimal	Secondary device coil current (AC).
WPT_EVPCCurrentOutputInformation	decimal	DC current supplied to the EV.
WPT_EVPCVoltageOutputInformation	decimal	DC bus or battery voltage.

2.17.4. ConnectedEV for ACDP

New in OCPP 2.1

The following parameters from ACDP_VehiclePositioningReq, ACPD_ConnectReq, ACDP_DisconnectReq and ACDP_SystemStatusReq can be represented in the ConnectedEV component. This information might be useful for troubleshooting, but is not required for normal operation.

Optional variables

Variable	Unit	ISO 15118-20 value
ACDP_EVMobilityStatus	boolean	Is true when an EV is immobilized
ACDP_EVPositioningSupport	boolean	Is true when EV has positioning support
ACDP_EVElectricalChargingDeviceStatus	OptionList	Values are: "State_A" (disconnected), "State_B", "State_C", "State_D" (connected)
ACDP_EVReadyToCharge	boolean	Element signalizes if the EV is READY or NOT READY to charge.
ACDP_EVImmobilizationRequest	boolean	Represents the request of immobilization of the EV. This may be related to the hand brake status.
ACDP_EVImmobilized	boolean	The immobilization of the EV is a mandatory precondition to activate the pantograph.
ACDP_WLANStrength	decimal	Element signalizes EV WiFi reception signal strength (-dBm)
ACDP_EVCPStatus	OptionList	Values are: "State_A", "State_B", "State_C", "State_D", "State_E". Refer to IEC 61851-23-1.
ACDP_EVSOC	decimal	For operation usable SOC status in %. This value may differ from physical SOC of the battery. This parameter can be same as PresentSOC in DisplayParameter.

Variable	Unit	ISO 15118-20 value
ACDP_EVErrorCode	OptionList	One of: "OK_NoEVError", "FAILED", "FAILED_EmergencyEvent", "FAILED_Breaker", "FAILED_RESSTemperatureInhibit", "FAILED_RESS", "FAILED_ChargingCurrentDifferential", "FAILED_ChargingVoltageOutOfRange", "FAILED_Reserved1", "FAILED_Reserved2"
ACDP_EVTimeout	boolean	Indicates the occurrence of a timeout in the EVCC.

2.18. Bidirectional Charging related

2.18.1. `V2XEnabled`

New in OCPP 2.1

Required	yes, when supporting V2X		
Component	**componentName**	V2XChargingCtrlr	
	evse	*	
Variable	**variableName**	Enabled	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean
Description	Way for the CSMS to either activate or deactivate V2X functionality on a Charging Station.		

2.18.2. `V2XSupportedEnergyTransferMode`

New in OCPP 2.1

Required	yes, when supporting V2X		
Component	**componentName**	V2XChargingCtrlr	
	evse	*	
Variable	**variableName**	SupportedEnergyTransferModes	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	Memberlist
		valuesList	values from EnergyTransferModeEnumType
		maxLimit	The maximum length of the values list, to be defined by the implementer.
Description	Lists the energy transfer services that are supported by the Charging Station.		

2.18.3. `V2XSupportedOperationModes`

New in OCPP 2.1

Required	yes, when supporting V2X	
Component	**componentName**	V2XChargingCtrlr
	evse	*

Variable	**variableName**	SupportedOperationModes	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	MemberList
		valuesList	values from OperationModeEnumType
		maxLimit	The maximum length of the values list, to be defined by the implementer.
Description	Lists the operation modes that are supported by the Charging Station.		

2.18.4. `V2XLocalFrequencyUpdateThreshold`

New in OCPP 2.1

Required	yes, when supporting V2X *operationMode* LocalFrequency		
Component	**componentName**	V2XChargingCtrlr	
	evse	*	
Variable	**variableName**	LocalFrequencyUpdateThreshold	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	integer
Description	The amount of change in net frequency in **mHz** is needed to trigger a recalculation of the setpoint. Typical values range from 1 to 100.		

2.18.5. V2X Additional Sampled Data Measurands

The following configuration variables define parameters that the Charging Station shall report via TransactionEventRequest messages **in addition to** those measurands that have been defined in the standard SampledDataCtrlr. The sampling interval for these additional measurands is configured in V2XChargingCtrlr.TxUpdatedInterval. When absent, the interval value from SampledDataCtrlr.TxUpdatedInterval is used.

2.18.5.1. `V2XSampledDataTxStartedMeasurands[OperationMode]`

New in OCPP 2.1

Required	no		
Component	**componentName**	V2XChargingCtrlr	
	evse	*	
Variable	**variableName**	TxStartedMeasurands	
	variableInstance	<OperationMode>	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	MemberList
		valuesList	values from MeasurandEnumType
Description	List of sampled measurands to send in addition to those configured in SampledDataCtrlr.TxStartedMeasurands, when the Charging Station is in V2X operation mode <OperationMode>. Measurands that occur in TxStartedMeasurands of both SampledDataCtrlr and in V2XChargingCtrlr are reported as one measurand.		

2.18.5.2. `V2XSampledDataTxEndedMeasurands[OperationMode]`

New in OCPP 2.1

Required	no		
Component	**componentName**	V2XChargingCtrlr	
	evse	*	
Variable	**variableName**	TxEndedMeasurands	
	variableInstance	<OperationMode>	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	MemberList
		valuesList	values from MeasurandEnumType
Description	List of sampled measurands to send in addition to SampledDataCtrlr.TxEndedMeasurands, when the Charging Station is in V2X operaion mode <OperationMode>. Measurands that occur in TxEndedMeasurands of both SampledDataCtrlr and in V2XChargingCtrlr are reported as one measurand.		

2.18.5.3. `V2XSampledDataTxEndedInterval[OperationMode]`

New in OCPP 2.1

Required	no		
Component	**componentName**	V2XChargingCtrlr	
	evse	*	
Variable	**variableName**	TxEndedInterval	
	variableInstance	<OperationMode>	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**unit**	s
		dataType	integer
Description	The interval used to sample the list of measurands in V2XChargingCtrlr.TxEndedMeasurands, while the Charging Station is in V2X operation mode <OperationMode>. When absent the interval defaults to that in SampledDataCtrlr.TxEndedInterval. Measurands that occur in TxEndedMeasurands of both SampledDataCtrlr and in V2XChargingCtrlr are reported as with the sampling interval of V2XChargingCtrlr.TxEndedInterval.		

2.18.5.4. `V2XSampledDataTxUpdatedMeasurands[OperationMode]`

New in OCPP 2.1

Required	no	
Component	**componentName**	V2XChargingCtrlr
	evse	*

Variable	**variableName**	TxUpdatedMeasurands	
	variableInstance	<OperationMode>	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	MemberList
		valuesList	values from MeasurandEnumType
Description	List of sampled measurands to send in addition to SampledDataCtrlr.TxUpdatedMeasurands, when the Charging Station is in V2X operation mode <OperationMode>. Measurands that occur in TxUpdatedMeasurands of both SampledDataCtrlr and in V2XChargingCtrlr are reported as one measurand.		

2.18.5.5. `V2XSampledDataTxUpdatedInterval[OperationMode]`

New in OCPP 2.1

Required	no		
Component	**componentName**	V2XChargingCtrlr	
	evse	*	
Variable	**variableName**	TxUpdatedInterval	
	variableInstance	<OperationMode>	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**unit**	s
		dataType	integer
Description	The interval used to sample the list of measurands in V2XChargingCtrlr.TxUpdatedMeasurands, while the Charging Station is in V2X operation mode <OperationMode>. When absent the interval defaults to that in SampledDataCtrlr.TxUpdatedInterval. Measurands that occur in TxUpdatedMeasurands of both SampledDataCtrlr and in V2XChargingCtrlr are reported as with the sampling interval of V2XChargingCtrlr.TxUpdatedInterval.		

2.18.6. V2X Local Load Balancing

The following variables control upper and lower threshold and offsets for local load balancing.

2.18.6.1. `V2XLocalLoadBalancing[UpperThreshold]`

New in OCPP 2.1

Required	yes, when supporting V2X *operationMode* LocalLoadBalancing		
Component	**componentName**	V2XChargingCtrlr	
	evse	*	
Variable	**variableName**	LocalLoadBalancing	
	variableInstance	UpperThreshold	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	decimal
Description	When power (in Watts) exceeds this limit, the local load balancing mechanism will try limit the power to a maximum of UpperThreshold + UpperOffset.		

2.18.6.2. `V2XLocalLoadBalancing[LowerThreshold]`

New in OCPP 2.1

Required	yes, when supporting V2X *operationMode* LocalLoadBalancing		
Component	**componentName**	V2XChargingCtrlr	
	evse	*	
Variable	**variableName**	LocalLoadBalancing	
	variableInstance	LowerThreshold	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	decimal
Description	When power (in Watts) drops below this limit, the local load balancing mechanism will try to keep the power to a minimum of LowerThreshold + LowerOffset.		

2.18.6.3. `V2XLocalLoadBalancing[UpperOffset]`

New in OCPP 2.1

Required	yes, when supporting V2X *operationMode* LocalLoadBalancing		
Component	**componentName**	V2XChargingCtrlr	
	evse	*	
Variable	**variableName**	LocalLoadBalancing	
	variableInstance	UpperOffset	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	decimal
Description	An offset value (in Watts) to add to the UpperThreshold. This allows for some tuning of the upper limit without changing the threshold value.		

2.18.6.4. `V2XLocalLoadBalancing[LowerOffset]`

New in OCPP 2.1

Required	yes, when supporting V2X *operationMode* LocalLoadBalancing		
Component	**componentName**	V2XChargingCtrlr	
	evse	*	
Variable	**variableName**	LocalLoadBalancing	
	variableInstance	LowerOffset	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	decimal
Description	An offset value (in Watts) to add to the LowerThreshold. This allows for some tuning of the lower limit without changing the threshold value.		

2.18.7. FrequencySimulator Component

The FrequencySimulator is an optional component that can simulate net frequencies during a certification process. It is a component that is reported as part of an EVSE and when enabled it only applies to that EVSE.

2.18.7.1. `FrequencySimulatorEnabled`

New in OCPP 2.1

Required	no		
Component	**componentName**	FrequencySimulator	
	evse	*	
Variable	**variableName**	Enabled	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean
Description	This variable must be set to true in order to activate the FrequencySimulator and gather data with the DataCollector. Sampling of data will occur between *DateTime["Start"]* and *DateTime["End"]*.		

2.18.7.2. `FrequencySimulatorDateTime[Start]`

New in OCPP 2.1

Required	no		
Component	**componentName**	FrequencySimulator	
	evse	*	
Variable	**variableName**	DateTime	
	variableInstance	Start	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	dateTime
Description	If *Enabled* the FrequencySimulator will simulate a net frequency as long as time >= *DateTime["Start"]* and time < *DateTime["End"]*.		

2.18.7.3. `FrequencySimulatorDateTime[End]`

New in OCPP 2.1

Required	no		
Component	**componentName**	FrequencySimulator	
	evse	*	
Variable	**variableName**	DateTime	
	variableInstance	End	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	dateTime
Description	If *Enabled* the FrequencySimulator will simulate a net frequency as long as time >= *DateTime["Start"]* and time < *DateTime["End"]*.		

2.18.7.4. `FrequencySimulatorSchedule`

New in OCPP 2.1

Required	no		
Component	**componentName**	FrequencySimulator	
	evse	*	
Variable	**variableName**	FrequencySchedule	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	string
Description	A JSON-formatted string with an array of { *time, freq* } pairs, in which *time* is time in seconds (as integer) relative to the beginning of the schedule at *DateTime("Start")* and *freq* is the frequency to be simulated during that period in Hz (as decimal). Behaviour is similar to that of a ChargingSchedulePeriodType, i.e. the start of a new period marks the end of the previous period and the final period lasts until *DateTime["End"]*. Example: "[{ "time": 0, "freq": 50.0 }, { "time": 60, "freq": 49.2 }, { "time": 300, "freq": 47.8 }]".		

2.18.8. DataCollector Component

The DataCollector is an optional component that collects samples of measurands at a high frequency, typically with subsecond intervals. It is a component that is reported as part of an EVSE and when enabled it only applies to that EVSE. The samples are logged internally in the Charging Station and can be retrieved in bulk with the GetLogRequest by specifying LogEnumType `DataCollectorLog`. For small intervals and lots of data it is much more efficient to use the DataCollector than installing a sampling monitor.

Its purpose is to collect high frequency samples (less than 100 ms intervals) that are needed during the certification process of a Charging Station that is doing local frequency control. The DataCollector component is, however, not specific to V2X operations and can be used for any sampling.

2.18.8.1. `DataCollectorEnabled`

New in OCPP 2.1

Required	no		
Component	**componentName**	DataCollector	
	evse	*	
Variable	**variableName**	Enabled	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	boolean
Description	This variable must be set to True in order to activate the DataCollector. Sampling of data will occur between *DateTime["Start"]* and *DateTime["End"]*.		

2.18.8.2. `DataCollectorDateTime[Start]`

New in OCPP 2.1

Required	no	
Component	**componentName**	DataCollector
	evse	*

Variable	**variableName**	DateTime	
	variableInstance	Start	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	dateTime
Description	If *Enabled* the DataCollector will start sampling measurands from *SampledMeasurands* as long as time >= *DateTime["Start"]* and time < *DateTime["End"]*.		

2.18.8.3. `DataCollectorDateTime[End]`

New in OCPP 2.1

Required	no		
Component	**componentName**	DataCollector	
	evse	*	
Variable	**variableName**	DateTime	
	variableInstance	End	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	dateTime
Description	If *Enabled* the DataCollector will start sampling measurands from *SampledMeasurands* as long as time >= *DateTime["Start"]* and time < *DateTime["End"]*.		

2.18.8.4. `DataCollectorSampledMeasurands`

New in OCPP 2.1

Required	no		
Component	**componentName**	DataCollector	
	evse	*	
Variable	**variableName**	SampledMeasurands	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	MemberList
		valuesList	values from MeasurandEnumType
		maxLimit	Maximum length of the memberlist as a CSV formatted string.
Description	The set of measurands to be sampled by the DataCollector. Samples are always appended to the `DataCollectorLog` until it is retrieved by a GetLogRequest(`DataCollectorLog`), after which it is cleared.		

2.18.8.5. `DataCollectorSamplingInterval`

New in OCPP 2.1

Required	no

Component	**componentName**	DataCollector	
	evse	*	
Variable	**variableName**	SamplingInterval	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**unit**	s
		dataType	decimal
Description	The sampling interval in **seconds**.		

2.19. DER Control related

2.19.1. DCDERCtrlr

New in OCPP 2.1

The DCDERCtrlr component holds DER capabilities of the DC inverter of the EVSE in the Charging Station. The component is located at the EVSE level, since it represents the DER capabilities, also referred to as nameplate information, of the EVSE.

Variable	Data Type	M/O	Description
MaxW	decimal	M	Active power rating in watts at unity power factor
OverExcitedW	decimal	M	Active power rating in watts at specified over-excited power factor
OverExcitedPF	decimal	M	Over-excited power factor
UnderExcitedW	decimal	M	Active power rating in watts at specified under-excited power factor
UnderExcitedPF	decimal	M	Under-excited power factor
MaxVA	decimal	M	Maximum apparent power rating in voltamperes
MaxVar	decimal	M	Maximum injected reactive power rating in vars
MaxVarNeg	decimal	M	Maximum absorbed reactive power rating in vars
MaxChargeRateW	decimal	M	Maximum active power charge rating in watts
MaxChargeRateVA	decimal	M	Maximum apparent power charge rating in voltamperes; may differ from the apparent power maximum rating
VNom	decimal	O	Nominal ac voltage rating in rms volts
MaxV	decimal	O	Maximum ac voltage rating in rms volts
MinV	decimal	O	Minimum ac voltage rating in rms volts
ModesSupported	MemberList of DERControlEnumType	M	Indication of support for each control mode function
InverterManufacturer	string	M	Manufacturer of inverter
InverterModel	string	M	Model of inverter
InverterSerialNumber	string	O	Serial number of inverter
InverterSwVersion	string	M	Software version of inverter
InverterHwVersion	string	M	Hardware version of inverter
IslandingDetectionMethod	OptionList of IslandingDetectionEnumType	O	Type of islanding detection method
IslandingDetectionTripTime	decimal	O	Time until tripping after island detection
ReactiveSusceptance	decimal	M	Reactive susceptance that remains connected to the electrical power system in the cease to energize and trip state

M/O = "mandatory when DER control supported" or "optional"

2.19.2. ACDERCtrlr

New in OCPP 2.1

The ACDERCtrlr component holds DER capabilities that the EVSE of the Charging Station can emulate by using ISO 15118-20 ChargeLoop messages to control the inverter in the EV. The component is located at the EVSE level, since it represents the DER capabilities of the EVSE.

If the *modesSupported* variable is an empty list, then all DER capabilities must be provided by the EV.

Variable	Data Type	M/O	Description
ModesSupported	MemberList of DERControlEnumType	M	Indication of support for each control mode function

M/O = "mandatory when DER control supported" or "optional"

2.20. Battery Swapping related

2.20.1. `BatterySwapTargetSoc`

New in OCPP 2.1

Required	no		
Component	**componentName**	BatterySwapCtrlr	
Variable	**variableName**	TargetSoc	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	integer
Description	The state of charge that a battery must have in order to be eligible for swapping. Batteries below TargetSoc are reported as "Occupied", at or above TargetSoc they are reported as "Available". TargetSoc must be smaller or equal to MaxSoc. A battery will continue to be charged until MaxSoc.		

2.20.2. `BatterySwapMaxSoc`

New in OCPP 2.1

Required	no		
Component	**componentName**	BatterySwapCtrlr	
Variable	**variableName**	MaxSoc	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	integer
Description	The maximum state of charge that a battery will be charged to.		

2.20.3. `BatterySwapIdtoken`

New in OCPP 2.1

Required	no	
Component	**componentName**	BatterySwapCtrlr

Variable	**variableName**	IdToken	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	string
Description	The idToken that is used for charging transactions of swap batteries. If absent or empty string, then *tokenType* = "NoAuthorization" is used, else the given value is used as *idToken* with *tokenType* = `Central`.		

2.20.4. `BatterySwapInTimeout`

New in OCPP 2.1

Required	no		
Component	**componentName**	BatterySwapCtrlr	
Variable	**variableName**	Timeout	
	variableInstance	In	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	integer
Description	Timeout in seconds in which a set of batteries must be inserted after successful authorization.		

2.20.5. `BatterySwapOutTimeout`

New in OCPP 2.1

Required	no		
Component	**componentName**	BatterySwapCtrlr	
Variable	**variableName**	Timeout	
	variableInstance	Out	
	variableAttributes	**mutability**	ReadWrite
	variableCharacteristics	**dataType**	integer
Description	Timeout in seconds in which the set of batteries that is offered by Charging Station to take out in exchange for the inserted set of batteries must be removed.		

2.20.6. `BatteryCartridgeSoC`

New in OCPP 2.1

Required	no		
Component	**componentName**	BatteryCartridge	
	evse	*	
Variable	**variableName**	SoC	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	integer

Description	The component BatteryCartridge refers to the battery that is inserted at the EVSE. The variable SoC represents current state of charge as a percentage from 0..100.

2.20.7. `BatteryCartridgeSoH`

New in OCPP 2.1

Required	no		
Component	**componentName**	BatteryCartridge	
	evse	*	
Variable	**variableName**	SoH	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	integer
Description	The component BatteryCartridge refers to the battery that is inserted at the EVSE. The variable SoH represents current state of health as a value from 0..100.		

2.20.8. `BatteryCartridgeWorkingMode`

New in OCPP 2.1

Required	no		
Component	**componentName**	BatteryCartridge	
	evse	*	
Variable	**variableName**	WorkingMode	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	OptionList
		valuesList	"CC, CV, Full, Ready, Standby"
Description	This variable represents the current working mode of the battery. CC: Continuous Current, CV: Continuous Voltage, Full: Fully Charged, Ready: Ready to be charged, Standby: Battery slot is empty This optional variable can be sent to or monitored by CSMS for information.		

2.20.9. `BatterySlotStatus`

New in OCPP 2.1

Required	no		
Component	**componentName**	Connector	
	evse	*	
Variable	**variableName**	SlotStatus	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	OptionList
		valuesList	"Open, Locked, Faulted"

Description	This variable represents the status of the door of the battery slot. This optional variable can be sent to or monitored by CSMS for information.

2.20.9.1. Example representation of a battery swap station in device model

A battery swap station can be mapped onto the standard 3-tier model of a charging station. A battery slot is mapped onto an EVSE. The battery slot type is represented by the connector type. Variables associated with the inserted battery cartridge are represented by a BatteryCartridge component, which is only Available when a cartridge is inserted. The table below represents a battery swap station with two battery slots in which only slot #2 contains a battery cartridge.

Component				**Variable**		**VariableAttribute**		**VariableCharacteristics**		
name	**evse id**	**evse connectorId**	**instance**	**name**	**instance**	**type**	**value**	**dataType**	**maxLimit**	**supports Monitoring**
ChargingStation				Available		Actual	true	boolean		false
ChargingStation				AvailabilityState		Actual	Available	boolean		true
BatterySwapCtrlr				TargetSoC		Actual	70	integer	100	false
BatterySwapCtrlr				MaxSoC		Actual	80	integer	100	false
BatterySwapCtrlr				IdToken		Actual	BSS-001	string		false
EVSE	1			Available		Actual	true	boolean		false
EVSE	1			AvailabilityState		Actual	Available	OptionList		true
EVSE	1			Power		Actual	0.0	decimal	22000.0	true
Connector	1	1		Available		Actual	true	boolean		false
Connector	1	1		ConnectorType		Actual	BatterySlot	string		false
Connector	1	1		SlotStatus		Actual	Open	OptionList		true
BatteryCartridge	1	1		Available		Actual	false	boolean		true
BatteryCartridge	1	1		Temperature		Actual	0	decimal		true
BatteryCartridge	1	1		SoC		Actual	0	integer		true
BatteryCartridge	1	1		WorkingMode		Actual	CC	OptionList		true
BatteryCartridge	1	1		SlotStatus		Actual	Locked	OptionList		true
EVSE	2			Available		Actual	true	boolean		false
EVSE	2			AvailabilityState		Actual	Occupied	optionList		true
EVSE	2			Power		Actual	4100.0	decimal	5000.0	true
Connector	2	1		Available		Actual	true	boolean		false
Connector	2	1		AvailabilityState		Actual	Occupied	optionList		true
Connector	2	1		ConnectorType		Actual	BatterySlot	string		false
Connector	1	1		SlotStatus		Actual	Locked	OptionList		true
BatteryCartridge	2	1		Available		Actual	true	boolean		true
BatteryCartridge	2	1		Temperature		Actual	35	decimal		true
BatteryCartridge	2	1		SoC		Actual	51	integer		true
BatteryCartridge	2	1		WorkingMode		Actual	CC	OptionList		true
BatteryCartridge	2	1		SlotStatus		Actual	Locked	OptionList		true

NOTE The BatteryCartridge component a EVSE #1 is absent. As a result the Available variable is false. The other variables (like Temperature, SoC) still exist in the device model, but have no meaning when the battery cartridge is not present.

2.21. Device Model hierarchy variables

New in OCPP 2.1

See Part 1 "Device Model Hierarchy" for the purpose of these variables.

The value of a XxxParent variable (Communication/Electrical/Logical/PhysicalParent) is a shorthand notation that points to the parent component, as follows:

```
"[EvseId[#ConnectorId]/]Component[#Instance]".
```

in which parts between square brackets "[]" are optional.

For example:

- To reference the Charging Station tier component "PowerBank" with instance "1", the variable value is **"PowerBank#1"**
- To reference the EVSE tier component on evseId #2 named "Meter" with instance "1" the variable value is **"2/Meter#1"**
- To reference the Connector tier component on evseId #2 and connectorId #1 named "Shutter" with no instance, the variable value is **"2#1/Shutter"**

2.21.1. CommunicationParent

Required	no		
Component	**componentName**	<any>	
Variable	**variableName**	CommunicationParent	
	variableInstance	<any> or none	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	string
Description	Points to a communication parent component (data flow source), to allow rendering the communication hierarchy in a UI. Value format: [EvseId[#ConnectorId]/]Component[#Instance]		

2.21.2. ElectricalParent

Required	no		
Component	**componentName**	<any>>	
Variable	**variableName**	ElectricalParent	
	variableInstance	<any> or none	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	string
Description	Points to a electrical parent component (energy flow source), to allow rendering the electrical hierarchy in a UI. Value format: [EvseId[#ConnectorId]/]Component[#Instance]		

2.21.3. LogicalParent

Required	no	
Component	**componentName**	<any>>

Variable	**variableName**	LogicalParent	
	variableInstance	<any> or none	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	string
Description	Points to a logical parent component, to allow rendering a comprehensive overview of the Charging Station components in a UI. Value format: [EvseId[#ConnectorId]/]Component[#Instance]		

2.21.4. PhysicalParent

Required	no		
Component	**componentName**	<any>>	
Variable	**variableName**	PhysicalParent	
	variableInstance	<any> or none	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	string
Description	Points to a physical parent component (container), to allow rendering an overview of the Charging Station component locations in a UI. Value format: [EvseId[#ConnectorId]/]Component[#Instance]		

2.21.5. Label

Required	no		
Component	**componentName**	<any>>	
Variable	**variableName**	Label	
	variableInstance	<any> or none	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	string
Description	Specifies a non-unique label to be used in a hierarchy UI rendering, in place of the unique component name and instance, in case a duplicate label is needed (e.g. when a component physically contained in a parent component has the same label as a component contained in another parent component).		