OCPP 2.0.1 Edition 3

Errata 2025-01

Table of Contents

Disclaimer

Copyright © 2010 – 2025 Open Charge Alliance. All rights reserved.

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

Version History

Version	Date	Description
2025-01	2025-01-23	Includes new errata for Part 1-4 of OCPP 2.0.1 Edition 3
2024-11	2024-11-14	Includes new errata for Part 5 and Part 6 of OCPP 2.0.1 Edition 3
2024-09	2024-09-25	Includes new errata for Part 4, Part 5 and Part 6 of OCPP 2.0.1 Edition 3
2024-06	2024-06-27	Includes new errata for Part 5 and Part 6.

Scope

This document contains errata on the OCPP 2.0.1 documentation. These errata have to be read as an addition to the release of OCPP 2.0.1 Edition 3.

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

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

Terminology and Conventions

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

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

This is version 2025-01 of the errata. The errata of this version are marked with "(2025-01)" in the section title.

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

0. Part 0

Currently no new errata for OCPP 2.0.1 Edition 3 part 0.

1. Part 1

1.1. Page 9 - (2025-01) - 3.4. Monitoring

The first sub-bullit is further clarified that it refers to the absolute difference:

	Description
Old	When the dataType of the variable is integer or decimal, this value represents the difference to be reached to trigger the monitor.
New	When the dataType of the variable is integer or decimal, this value represents the **absolute** difference to be reached to trigger the monitor.

2. Part 2

2.1. Page 5 - (2025-01) - Updated limitations on BasicAuthPassword to increase security

A "passwordString" should not be limited to any set of symbols as this limits its security greatly. There is also no reason for limiting the password size.

Table 1. Primitive Datatypes

Datatype	Description
passwordString	This is a UTF-8 encoded case-sensitive string. ~~that can only contain characters from the following character set: "a-z", "A-Z", "0-9" or any of the following limited set of symbols: * - _ = : + | @ .~~

	ID	Precondition	Requirement definition
Old text	A00.FR.205		The password SHALL be stored in the BasicAuthPassword configuration variable. It SHALL be a randomly chosen passwordString with a sufficiently high entropy, consisting of minimum 16 and maximum 40 characters (alpha-numeric characters and the special characters allowed by passwordString). The password SHALL be sent as a UTF-8 encoded string (NOT encoded into octet string or base64).
New text	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).
Old text	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 maximum 40 characters (alpha-numeric characters and the special characters allowed by passwordString). The password SHALL be sent as a UTF-8 encoded string (NOT encoded into octet string or base64). (Same as A00.FR.205)
New text	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)

2.1.1. BasicAuthPassword

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

Required	no		
Component	**componentName**	SecurityCtrlr	
Variable	**variableName**	BasicAuthPassword	
	variableAttributes	**mutability**	WriteOnly
	variableCharacteristics	**dataType**	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.

2.2. Page 15 - (2025-01) - Improved text FR.04

The requirement did not take into account the pending status and should refer to the applicable use cases B02 and B03.

	ID	Precondition	Requirement definition	Note
Old text	FR.04	When the CSMS did not accept the BootNotificationRequest from the Charging Station AND The Charging Station sends a message other than BootNotificationRequest	The CSMS SHALL respond with a RPC Framework: CALLERROR: SecurityError.	
New text	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.

2.3. Page 19 - (2025-01) - Removed requirement A00.FR.003 as the precondition never occurs

Requirement A00.FR.003 describes a precondition that never occurs in which the CSMS connects to the Charging Station. The requirement is therefore removed.

ID	Precondition	Requirement definition
~~A00.FR.003~~	~~If the CSMS tries to connect with a different profile than the Charging Station is using~~	~~The Charging Station SHALL terminate the connection.~~

2.4. Page 19/41 - (2025-01) - Allow downgrading security profile from 3 to 2

For migration purposes a Charging Station implementer has now the option for their Charging Stations to allow downgrading the security profile from 3 to 2, because not all CSMSs support security profile 3. ENCS was consulted to identify if this would pose any security risks. This was not the case as security profile 2 still uses TLS, which provides sufficient security.

Table 2. A05 - Upgrade Charging Station Security Profile

No.	Type	Description
1	**Name**	Upgrade Charging Station Security Profile
2	**ID**	A05
	Functional block	A. Security
3	**Objective(s)**	The CSO wants to ~~increase~~ 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 2.0.1. Before migrating to a security profile, the prerequisites, like installed certificates or password, need to be configured.
	Actors	CSMS, Charging Station

No.	Type	Description
	Scenario description	**1.** The CSMS sets a new value for the `NetworkConfigurationPriority` Configuration Variable via SetVariablesRequest, such that the NetworkConnectionProfile for the new ~~(higher)~~ 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 ~~(higher)~~ an allowed security profile AND that the prerequisite(s) for going to ~~a higher~~ the new security profile are met before sending the command to change to ~~a higher~~ the new security profile.
6	**Postcondition(s)**	The Charging Station was successfully upgraded to a ~~higher~~ 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.

Changed A00 requirement:

	ID	Precondition	Requirement definition
Old text	A00.FR.005		Lowering the security profile that is used, to a less secure profile, is for security reasons, not part of the OCPP specification, and MUST be done through another method, not via OCPP. OCPP messages SHALL NOT be used for this (e.g. SetVariablesRequest or DataTransferRequest).
New text	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.

Added A05 requirements:

	ID	Precondition	Requirement definition
New requirement	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)

	ID	Precondition	Requirement definition
New requirement	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)
New requirement	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)

Changed/added B09 requirements:

	ID	Precondition	Requirement definition
Old text	B09.FR.04	On receipt of the SetNetworkProfileRequest AND the NetworkConnectionProfile contains a lower securityProfile than stored at the configuration variable `SecurityProfile`	The Charging Station SHALL respond by sending a SetNetworkProfileResponse message, with status *Rejected*
New text	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*
New requirement	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
New requirement	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)

Added referenced variable:

2.4.1. 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.		

2.5. Page 26 - (2025-01) - 1.3.7. TLS with Client Side Certificates Profile - Requirements

A new requirement is added to support A00.FR.429:

New requirement

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

ID	Precondition	Requirement definition
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.

2.6. Page 28 - (2025-01) - Certificate Hierarchy

A note has been added to the specification to warn OCPP implementers about the potential risks involved with the installation of CA bundles.

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.

2.7. Page 31 - (2025-01) - A02/A03 - Updated error handling

NOT RECOMMENDED in the error handling is confusing as there is already a requirement that explicitly states that the Charging Station SHALL retry after "CertSigningWaitMinimum".

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. ~~It is NOT RECOMMENDED to let the Charging Station retry when the certificate is not send within X minutes or hours. When the CSR is incorrect, that will not be resolved automatically. It is possible that only a new firmware will fix this.~~

2.8. Page 34 - (2025-01) - Disallow client certificates future validity date

Currently the specification is unclear on whether accepting client certificates with a future validity date is intended behavior. If it would be allowed behavior, several problems were identified. For example, there would be no way for the CSMS to manage these (ghost) client certificates using OCPP. ENCS was consulted if there would be any security risks when not allowing accepting client certificates with a future validity date. ENCS identified no security risks, therefore it was decided to not allow accepting client certificates with a future validity date. In addition clarifications have been made to describe how to handle non-happy flow scenarios in which the Charging Station is unable to successfully connect using the new client certificate in combination with the NetworkConnectionProfile mechanism described at use case B10.

Changed/added A02 requirements:

	ID	Precondition	Requirement definition	Note
Old text	A02.FR.06		The Charging Station SHALL verify the validity of the signed certificate in the CertificateSignedRequest message, checking at least the period when the certificate is valid, the properties in Certificate Properties, and that it is part of the Charging Station Operator certificate hierarchy as described in Certificate Hierarchy.	
New text	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.
Old text	A02.FR.08		The Charging Station SHALL switch to the new certificate as soon as the current date and time is after the 'Not valid before' field in the certificate (e.g. by closing the websocket and TLS connection and reconnecting with the new certificate).	
New text	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.	
Old text	A02.FR.09	If the Charging Station contains more than one valid certificate of the *ChargingStationCertificate* type.	The Charging Station SHALL use the newest certificate, as measured by the start of the validity period.	
New text	A02.FR.09		<Requirement removed>	

	ID	Precondition	Requirement definition	Note
Old text	A02.FR.10	A02.FR.09 AND When the Charging Station has validated that the new certificate works	The Charging Station MAY discard the old certificate. It is RECOMMENDED to store old certificates for one month, as fallback.	
New text	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.
New requirement	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.
New requirement	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.

Changed/added A03 requirements:

	ID	Precondition	Requirement definition	Note
Old text	A03.FR.06		The Charging Station SHALL verify the validity of the signed certificate in the CertificateSignedRequest message, checking at least the period when the certificate is valid, the properties in Certificate Properties, and that it is part of the Charging Station Operator certificate hierarchy as described in Certificate Hierarchy. (Same as A02.FR.06)	
New text	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.

	ID	Precondition	Requirement definition	Note
Old text	A03.FR.08		The Charging Station SHALL switch to the new certificate as soon as the current date and time is after the 'Not valid before' field in the certificate (e.g. by closing the websocket and TLS connection and reconnecting with the new certificate). (Same as A02.FR.08)	
New text	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)	
Old text	A03.FR.09	If the Charging Station contains more than one valid certificate of the *ChargingStationCertificate* type.	The Charging Station SHALL use the newest certificate, as measured by the start of the validity period. (Same as A02.FR.09)	
New text	A03.FR.09		<Requirement removed>	
Old text	A03.FR.10	A03.FR09 AND When the Charging Station has validated that the new certificate works	The Charging Station MAY discard the old certificate. It is RECOMMENDED to store old certificates for one month, as fallback. (Same as A02.FR.10)	
New text	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.
New requirement	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.
New requirement	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.

Changed/added B10 requirements:

	ID	Precondition	Requirement definition	Note
Old text	B10.FR.07	B10.FR.03 AND All `NetworkProfileConnectionAttempts` for every entry of `NetworkConfigurationPriority` failed.	The Charging Station SHOULD fallback and start 'reconnecting' to the NetworkConnectionProfile for which the last successful connection was made.	'reconnecting' in this requirement, refers to the reconnection mechanism described at section 5.3. Reconnecting from "Part 4 - JSON over WebSockets implementation guide".

	ID	Precondition	Requirement definition	Note
New text	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".
New requirement	B10.FR.08	B10.FR.09	The Charging Station SHOULD restart connecting with all configured entries of the `NetworkConfigurationPriority`	
New requirement	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 either one or both reconnecting mechanisms described at requirements; B10.FR.07 and B10.FR.08.	This is to prevent the Charging Station from becoming a stranded asset.

2.9. Page 58/60 - (2025-01) - Missing requirement information about omitting the value for WriteOnly variables

The specification does describe already that the value for WriteOnly variables need to be omitted, however this information is not part of a requirement yet.

Changed B07 requirement:

	ID	Precondition	Requirement definition	Note
Old text	B07.FR.03	B07.FR.01	The Charging Station SHALL send the requested information 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.
New text	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.

Changed B08 requirement:

	ID	Precondition	Requirement definition
Old text	B08.FR.03	B08.FR.01	The Charging Station SHALL send the requested information via one or more NotifyReportRequest messages to the CSMS.
New text	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.

2.10. Page 69 - (2025-01) - B11 - Reset without Ongoing Transaction - Requirements

The definition of B11.FR.06 has been improved:

	ID	Precondition	Requirement definition
Old text	B11.FR.06	B11.FR.01 AND For example there is a firmware update ongoing that cannot be interrupted.	The Charging Station SHALL respond with a status Rejected.

	ID	Precondition	Requirement definition
New text	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*.

2.11. Page 73 - (2025-01) - B12 - Reset with Ongoing Transaction - Requirements

The requirement B12.FR.04, 08 and added B12.FR.10 have been improved:

	ID	Precondition	Requirement definition
Old text	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.
New text	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.
Old text	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.
New text	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.
Old text	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`
New text	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 *Rejected*

2.12. Page 77 - (2025-01) - 1.6 Relationship between authorization and transaction

A new section has been added after 1.5 Unknown Offline Authorization

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.

2.13. Page 80 - (2025-01) - C01 - EV Driver Authorization using RFID - Requirements

A requirement has been added to define when two *idTokens* are considered equal:

New requirement

ID	Precondition	Requirement definition	Note
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.

2.14. Page 80 - (2025-01) - C01 - EV Driver Authorization using RFID - Requirements

A requirement has been added to make explicit that authorization ends after EVConnectionTimeout:

New requirement

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.

2.15. Page 99 - (2025-01) - C09- Authorization by GroupId - Requirements

A GroupIdToken is controlled by CSMS. This should be reflected in the type.

New requirement

ID	Precondition	Requirement definition
C09.FR.13		The field *idToken.type* of a GroupIdToken SHOULD be `Central`

2.16. Page 101 - (2025-01) - Updated requirements related to clarify the relation between AuthCacheLifeTime and cacheExpiryDateTime

The OCPP specification already describes the relation between AuthCacheLifeTime and cacheExpiryDateTime at section 1.3 Authorization Cache, however the requirements are missing this information.

Changed C10 requirements:

	ID	Precondition	Requirement definition	Note
Old text	C10.FR.10	When the validity of an Authorization Cache entry expires.	The Authorization Cache entry SHALL be removed from the cache or changed to `Expired`.	
New text	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.
Old text	C10.FR.13	When IdTokenInfoType contains a value for *cacheExpiryDateTime*	The time a token is considered to be present in the cache is determined by *cacheExpiryDateTime*. This variable indicates the date and time after which a token expires in the Authorization Cache.	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).
New text	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).

2.17. Page 113 - (2025-01) - C16 - Stop Transaction with a Master Pass - Requirements

Requirements about actual stopping of transactions have been added/updated:

Updated requirement

Table 4. C16 - Stop Transaction with a Master Pass - Requirements

	ID	Precondition	Requirement definition
Old text	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.
New text	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.
Old text	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.
New text	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.

New requirements

ID	Precondition	Requirement definition
C16.FR.07	C16.FR.01 OR C16.FR.02	Charging Station SHALL stop the transaction as described in use case E07.
C16.FR.07	C16.FR.07	Charging Station SHALL set *transactionInfo.stoppedReason* = `MasterPass` in TransactionEventRequest with *eventType* = `Ended`.

2.18. Page 129 - (2025-01) - Updated sequence diagram E01 S5

After all added clarifications on the PowerPathClosed TxStartPoint the corresponding sequence diagram was not updated to reflect this.

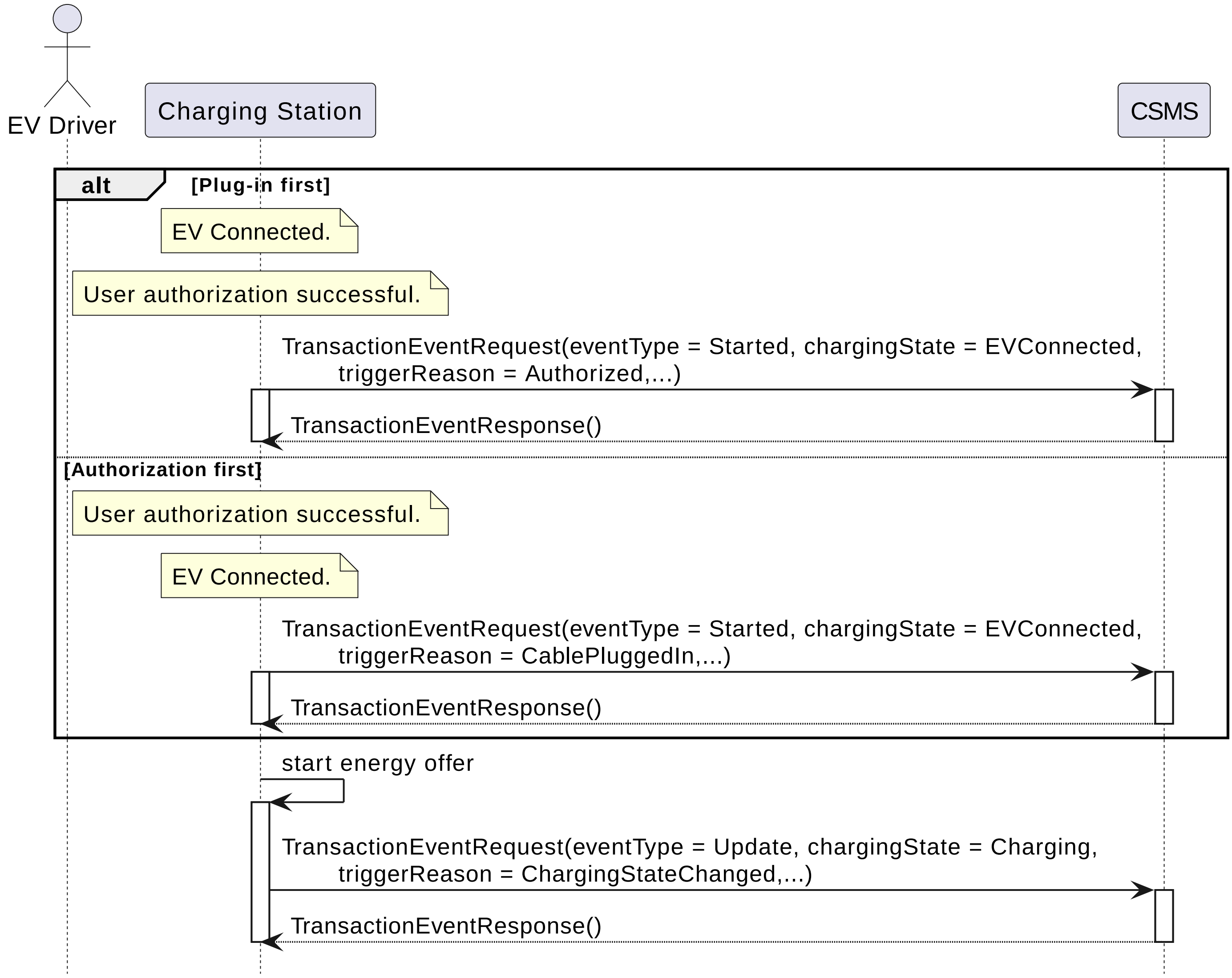
EV Driver
Charging Station
CSMS
alt
[Plug-in first]
EV Connected.
User authorization successful.
TransactionEventRequest(eventType = Started, chargingState = EVConnected, triggerReason = Authorized,...)
TransactionEventResponse()
[Authorization first]
User authorization successful.
EV Connected.
TransactionEventRequest(eventType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

Figure 1. Sequence Diagram: Start Transaction options - PowerPathClosed

2.19. Page 152 - (2025-01) - E07 - Improved scenario description names

The use case did not clearly indicate when the scenario and the alternative scenario are applicable.

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.

No.	Type	Description
	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.
...		

2.20. Page 154 - (2025-01) - E07 - Transaction locally stopped by IdToken

Sequence diagram Fig. 56 is updated to fix an error.
The fourth TransactionEventRequest needs to have: triggerReason = ChargingStateChanged, chargingState = EVConnected

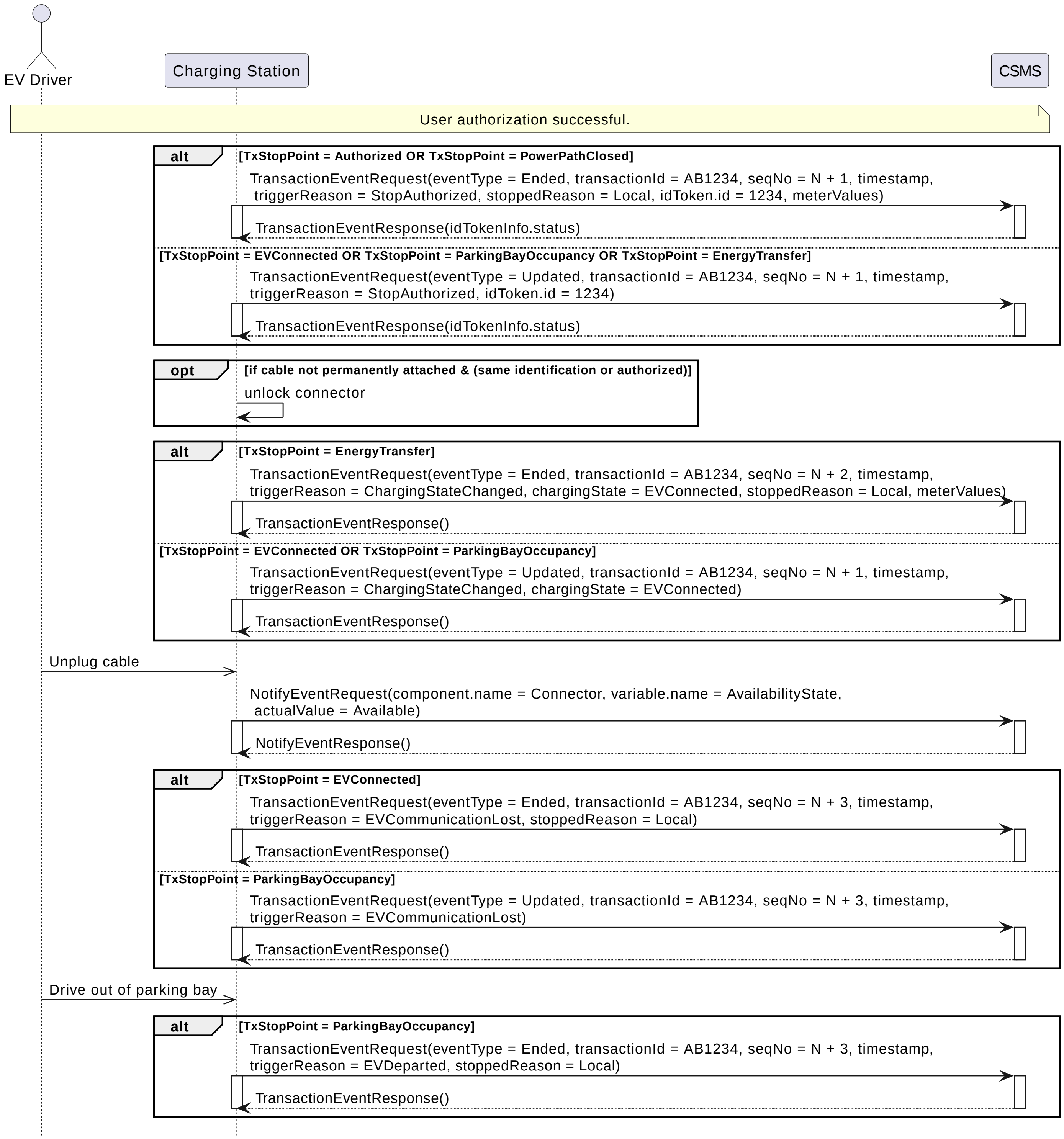
EV Driver
Charging Station
CSMS
User authorization successful.
alt
[TxStopPoint = Authorized OR TxStopPoint = PowerPathClosed]
TransactionEventRequest(eventType = Ended, transactionId = AB1234, seqNo = N + 1, timestamp, triggerReason = StopAuthorized, stoppedReason = Local, idToken.id = 1234, meterValues)
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 2. (Figure 56) Sequence Diagram: Transaction locally stopped by IdToken with TransactionEventRequest reported strictly by TxStopPoint configuration

2.21. Page 198 - (2025-01) - G01 - Status Notification - Requirements

A requirement is added to make explicit that a plug-in on reserved connector does not automatically change status to Occupied.

ID	Precondition	Requirement definition
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.

2.22. Page 197 - (2025-01) - G01 - Status Notification - State transition overview for connecting/disconnecting

The following text from the table has to be removed:

Initial	Cable plugin	Cable unplug
Available	→ Occupied	-
Occupied	-	→ Available (→ Unavailable, if scheduled to become Unavailable)
Reserved	(~~→ Occupied, only if authorized for reserved IdToken)~~	-
Unavailable	-	-
Faulted	-	-

2.23. Page 208 - (2025-01) - H. Reservation - Introduction

The Introduction text has been updated:"

Old text	This Functional Block describes the reservation functionality of OCPP. The reservation functionality enables an EV Driver to make a reservation of a Charging Station/EVSE, ensuring an available Connector at a Charging Station when he arrives. With Charging Stations not being abundantly available, and EVs having limited range, EV Drivers plan their trips from Charging Station to Charging Station. They need to know for sure they can use a Charging Station they plan to go to. They don’t like it when another EV Driver has started using the Charging Station in the time they were traveling to the Charging Station. For the EV Driver it is useful to be able to reserve a specific Type of Connector, or, when the EV Driver has no preference, an unspecified EVSE at a Charging Station. So he knows for sure he can charge at the Charging Station when he arrives.
New text	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.

2.24. Page 214/215 - (2025-01) - Improved use case scenario descriptions and added S3

The scenario descriptions have been updated to better explain the actual claiming of a reservation and have been clearly divided based on the configured TxStartPoint(s). In addition, A S3 has been added: '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.

No.	Type	Description
1	**Name**	Use a reserved EVSE

No.	Type	Description
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

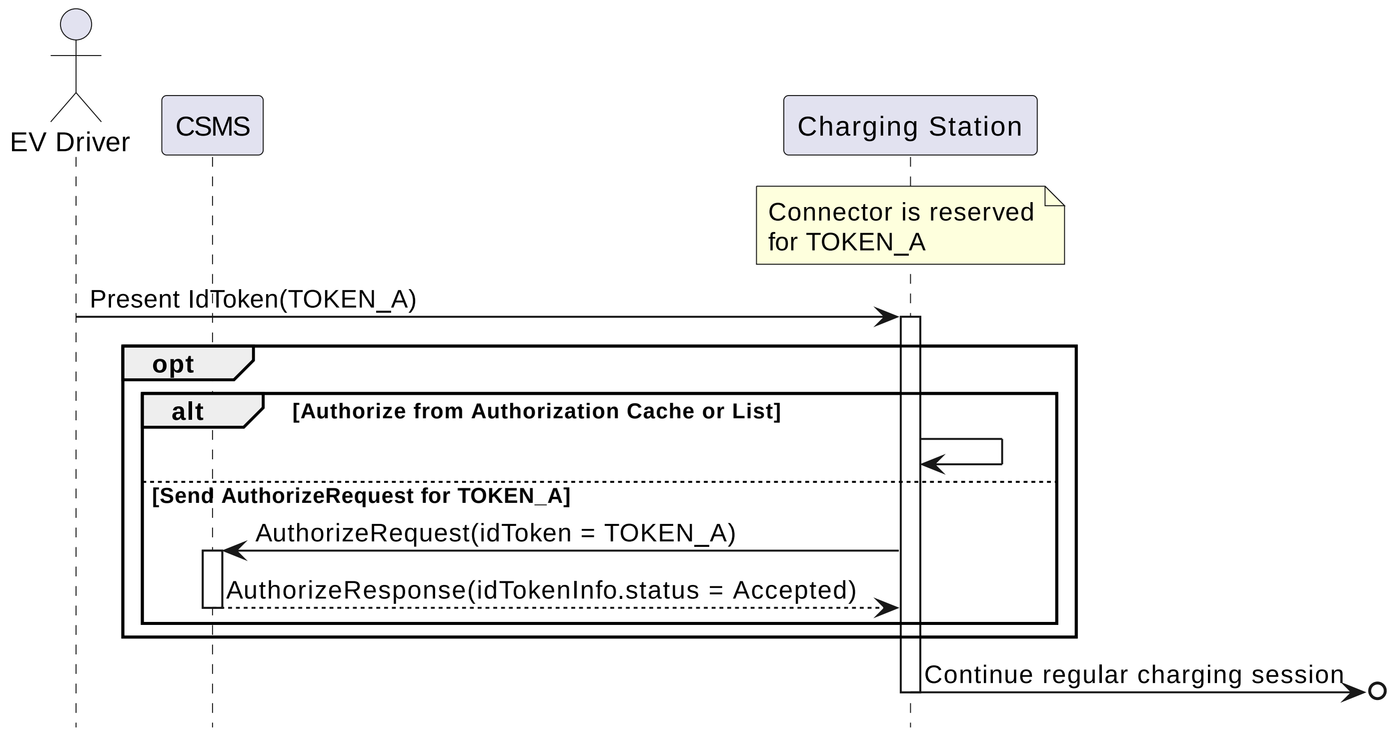
EV Driver
CSMS
Charging Station
Connector is reserved for TOKEN_A
Present IdToken(TOKEN_A)
opt
alt
[Authorize from Authorization Cache or List]
[Send AuthorizeRequest for TOKEN_A]
AuthorizeRequest(idToken = TOKEN_A)
AuthorizeResponse(idTokenInfo.status = Accepted)
Continue regular charging session

Figure 3. 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

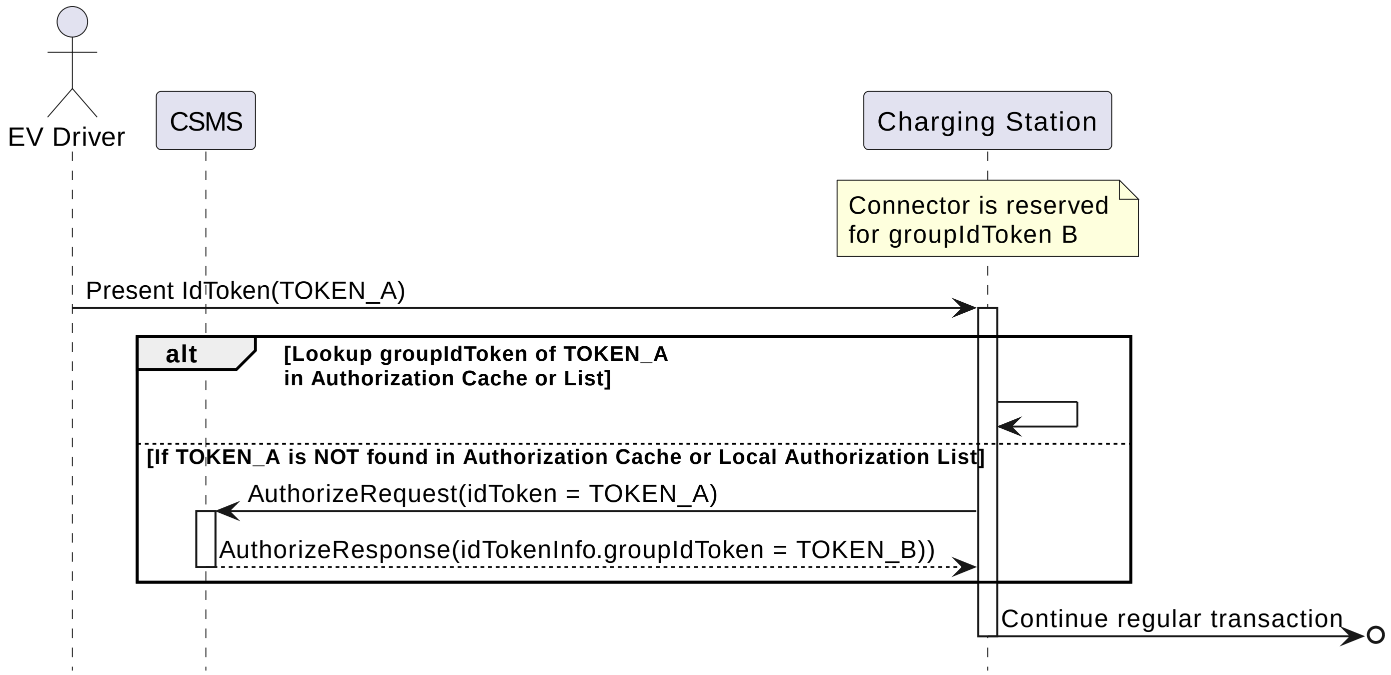
EV Driver
CSMS
Charging Station
Connector is reserved for groupIdToken B
Present IdToken(TOKEN_A)
alt
[Lookup groupIdToken of TOKEN_A in Authorization Cache or List]
[If TOKEN_A is NOT found in Authorization Cache or Local Authorization List]
AuthorizeRequest(idToken = TOKEN_A)
AuthorizeResponse(idTokenInfo.groupIdToken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

Figure 4. 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.

Added note:

	ID	Precondition	Requirement definition	Note
Old text	H03.FR.09	When an *idToken* or *groupIdToken* is presented that matches a reservation	Charging Station SHALL consider the reservation to be used (consumed)	
New text	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.

2.25. Page 226 - (2025-01) - I06.FR.02 Improved requirement text

Old	I06.FR.02	I06.FR.01 AND When there is updated tariff information available.	The CSMS SHALL respond with a TransactionEventResponse message to the Charging Station, containing the updated tariff information in the PersonalMessage field.
New	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 *updatedPersonalMessage* field.

2.26. Page 231 - (2025-01) - Updated section Multiple Locations/Phases

The section now specifies the 'relevant' configuration variables that apply.

Old	When a Charging Station can measure the same measurand on multiple locations or phases, all possible locations and/or phases SHALL be reported when configured in one of the relevant Configuration Variables.
New	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.

2.27. Page 243 - (2025-01) - Improved section on external Smart Charging Control Signals

To be inline with the EMS scenario additions to OCPP 2.1, this section also has been improved for OCPP 2.0.1.

Old text:

The OCPP protocol is originally developed for communication between a CSMS and one or more Charging Stations. As described in the above, this means that a Charging Station Operator (CSO) CSMS controls a Charging Station and, based on the charging limits of both the EV and the Charging Station, the CSO determines how fast the EV is charged. However, in some situations / applications of OCPP enabled Charging Stations, these are not the only 2 factors that determine the charging speed. Other inputs that determine charging speed could be DSO signals (e.g. via IEC 61850 [IEC61850-7-420], IEC 60870 [IEC60870-5-104], DNP3 [DNP3] or OpenADR [OPENADR]) or signals from a Building / Home Energy Management System. Although these signals are out of scope for OCPP, it seems clear from an OCPP perspective that the CSMS is to be informed of changes in charging by external signals. However, this also leads to a number of questions, such as how to deal with conflicting signals. The figure below presents an example setup with an Energy Management System, where the external signals are visualized both in a setup with direct communication to the Charging Station as well as a multiple Charging Station setup using a Local Controller:

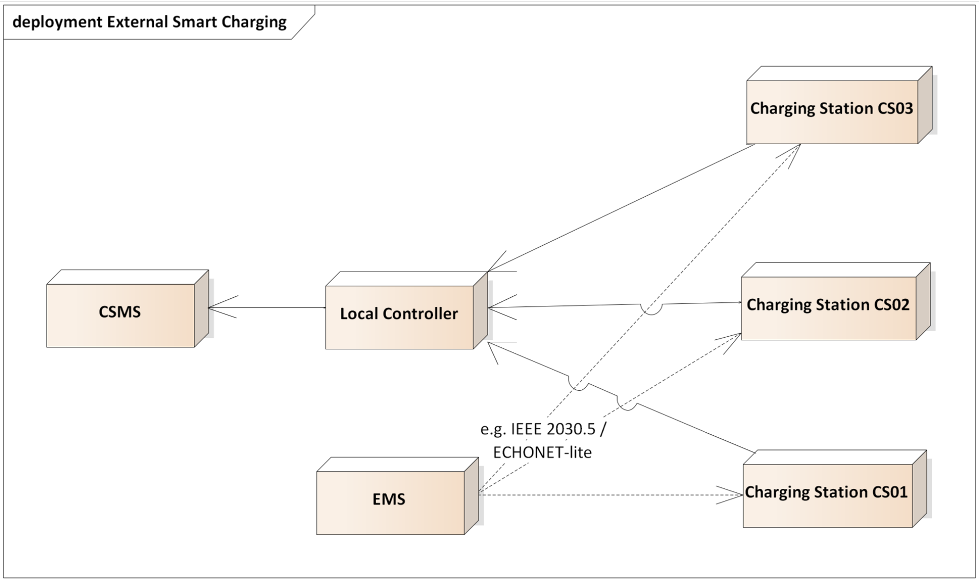

Figure 5. External Smart Charging

New text:

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.

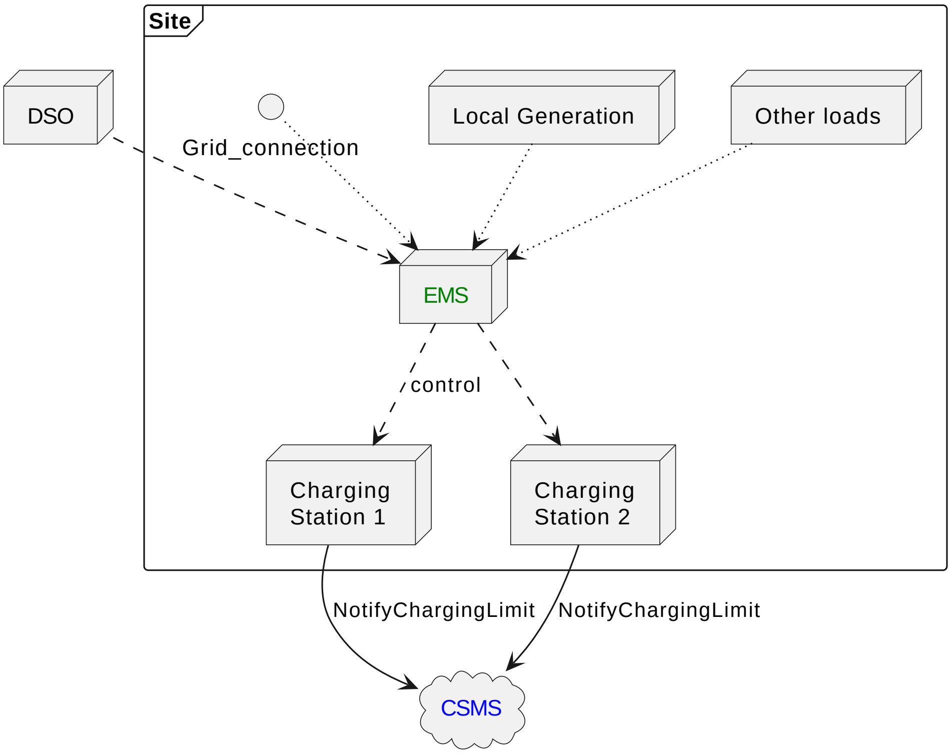

Figure 6. EMS control directly to Charging Stations

2.28. Page 248 - (2025-01) - 3.7 Avoiding Phase Conflicts

The following paragraph about Avoiding Phase Conflicts has been added:

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* = 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.

2.29. Page 286 - (2025-01) - K15 - ISO 15118-2 Charging with load leveling - Requirements

Added recommendation for timestamp when offline:

New / Updated requirements

	ID	Precondition	Requirements	Note
Changed	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
New	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.
New	K15.FR.21	K15.FR.10	Charging Station SHOULD set *selectedScheduleTupleId* to the Id of the *chargingSchedule* that EV selected from the provided ChargingProfileType(s).	

2.30. Page 286 - (2025-01) - K16 - Renegotiation initiated by CSMS - Requirements

The following requirement was added:

New requirement

ID	Precondition	Requirements	NOTE
K16.FR.14	K16.FR.05	Charging Station SHOULD set *selectedScheduleTupleId* to the Id of the *chargingSchedule* that EV selected from the provided ChargingProfileType(s).	

2.31. Page 292 - (2025-01) - Use case L01 - Added clarification to step 3 about when to start downloading the firmware

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). ...

2.32. Page 259/260 - (2025-01) - K03 - Updated use case description and sequence diagram

The use case description and sequence diagram have been updated to provide more information on how local load-balancing can be performed. (Requirements for K03 have not changed).

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

No.	Type	Description
	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**	
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.

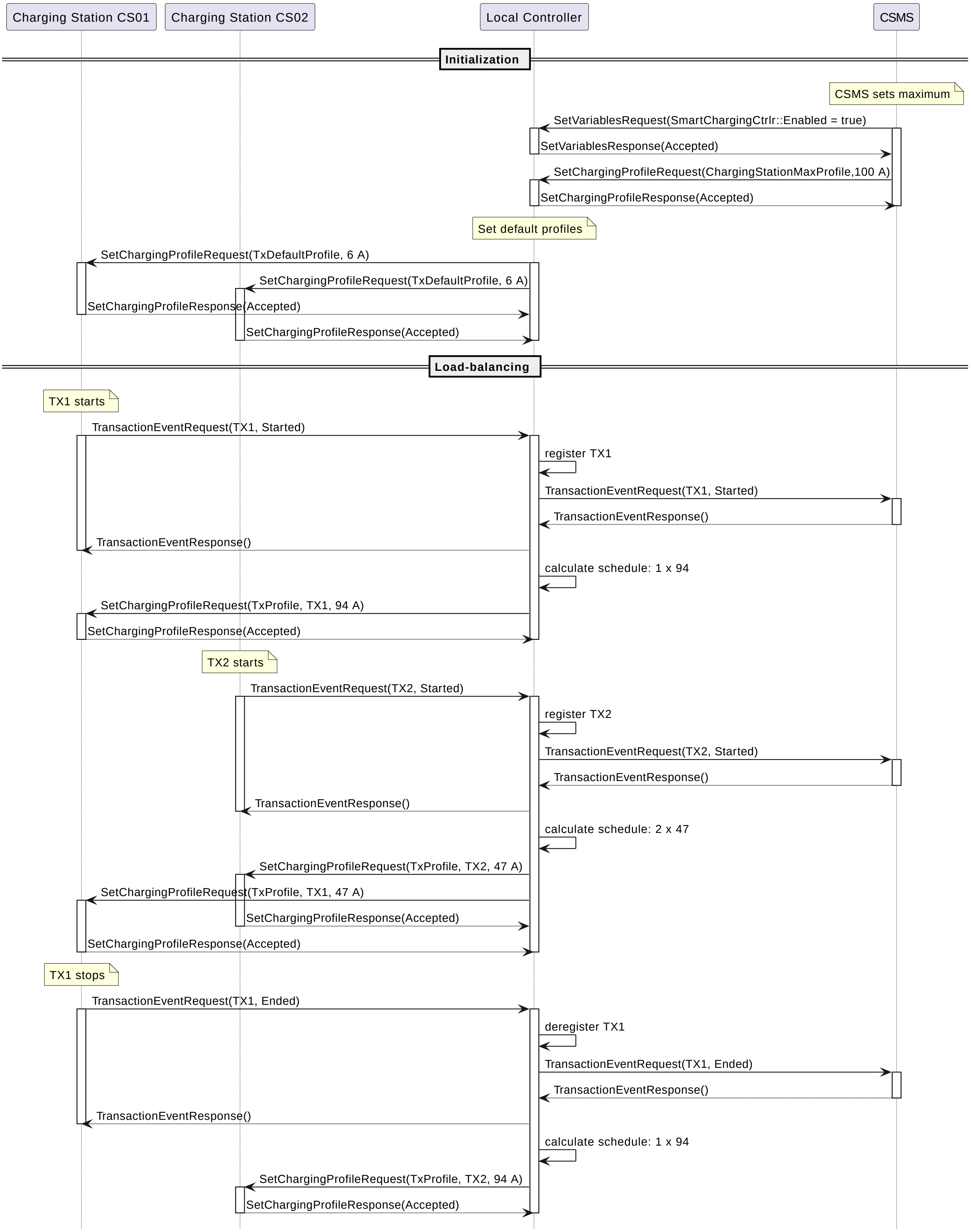
Charging Station CS01
Charging Station CS02
Local Controller
CSMS
Initialization
CSMS sets maximum
SetVariablesRequest(SmartChargingCtrlr::Enabled = true)
SetVariablesResponse(Accepted)
SetChargingProfileRequest(ChargingStationMaxProfile,100 A)
SetChargingProfileResponse(Accepted)
Set default profiles
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 7. Local Controller performing local load-balancing

2.33. Page 274/275/276/277 - (2025-01) - K11/K12 - Updated use case descriptions and sequence diagrams

The use case descriptions and sequence diagrams have been updated to describe the more likely scenario of a smart meter or EMS as external actor.

2.33.1. K11 - Set / Update External Charging Limit With Ongoing Transaction

No.	Type	Description
1	**Name**	Set / Update External Charging Limit With Ongoing Transaction
2	**ID**	K11
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.
...		

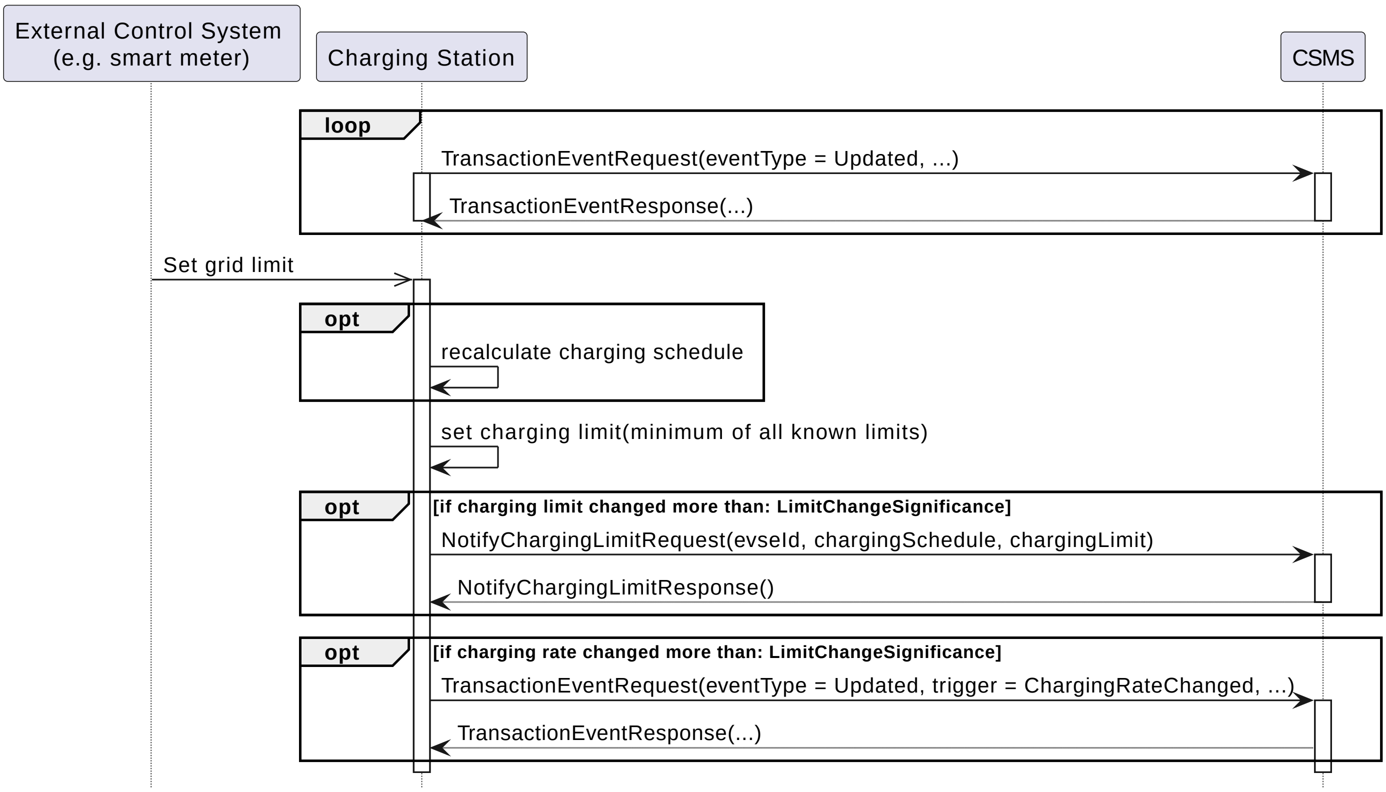
External Control System (e.g. smart meter)
Charging Station
CSMS
loop
TransactionEventRequest(eventType = Updated, ...)
TransactionEventResponse(...)
Set grid limit
opt
recalculate charging schedule
set charging limit(minimum of all known limits)
opt
[if charging limit changed more than: LimitChangeSignificance]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

Figure 8. Sequence diagram of the use case "Setting / Updating External Charging Limit with Ongoing Transaction"

2.33.2. K12 - Set / Update External Charging Limit Without Ongoing Transaction

No.	Type	Description
1	**Name**	Set / Update External Charging Limit Without Ongoing Transaction

No.	Type	Description
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.
…		

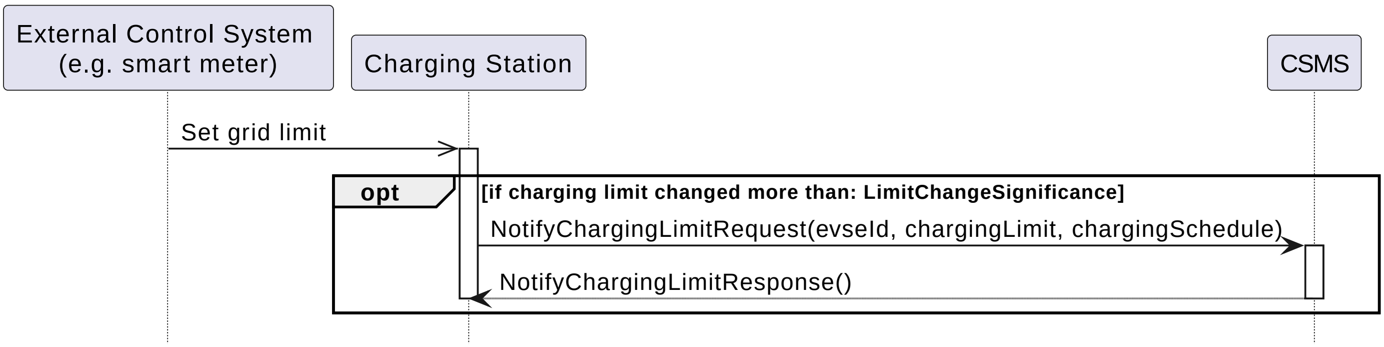
External Control System (e.g. smart meter)
Charging Station
CSMS
Set grid limit
opt
[if charging limit changed more than: LimitChangeSignificance]
NotifyChargingLimitRequest(evseId, chargingLimit, chargingSchedule)
NotifyChargingLimitResponse()

Figure 9. Sequence diagram of the use case "Set / Update External Charging Limit Without Ongoing Transaction"

2.34. Page 278/279 - (2025-01) - K13 - Updated requirement preconditions

Moved precondition "transaction is ongoing" from K13.FR.01 to K13.FR.03.

Changed K13 requirements:

	ID	Precondition	Requirements
Old text	K13.FR.01	~~A transaction is ongoing~~ ~~AND~~ External charging limit is released/removed	The Charging Station SHALL NOT limit charging anymore based on the previously received limit.
New text	K13.FR.01	External charging limit is released/removed	The Charging Station SHALL NOT limit charging anymore based on the previously received limit.
Old text	K13.FR.03	K13.FR.01 AND Charging rate changed by more than: `LimitChangeSignificance`	The Charging Station SHALL send a TransactionEventRequest message to the CSMS with trigger = ChargingRateChanged.

	ID	Precondition	Requirements
New text	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.

2.35. Page 279 - (2025-01) - K14 - Updated use case scenario description

Improved scenario step description 5 and 7.

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.
...		

2.36. Page 306 - (2025-01) - M. ISO 15118 Certificate Management

The functional block ISO 15118 Certificate Management is renamed to Certificate Management since this also contains certificate management of non-ISO 15118 related certificates.

2.37. Page 308 - (2025-01) - Update introduction sequence diagram ISO 15118

Not all ISO 15118 message sequences were complete, so the sequence diagram has been updated to show a more complete version of the message flow.

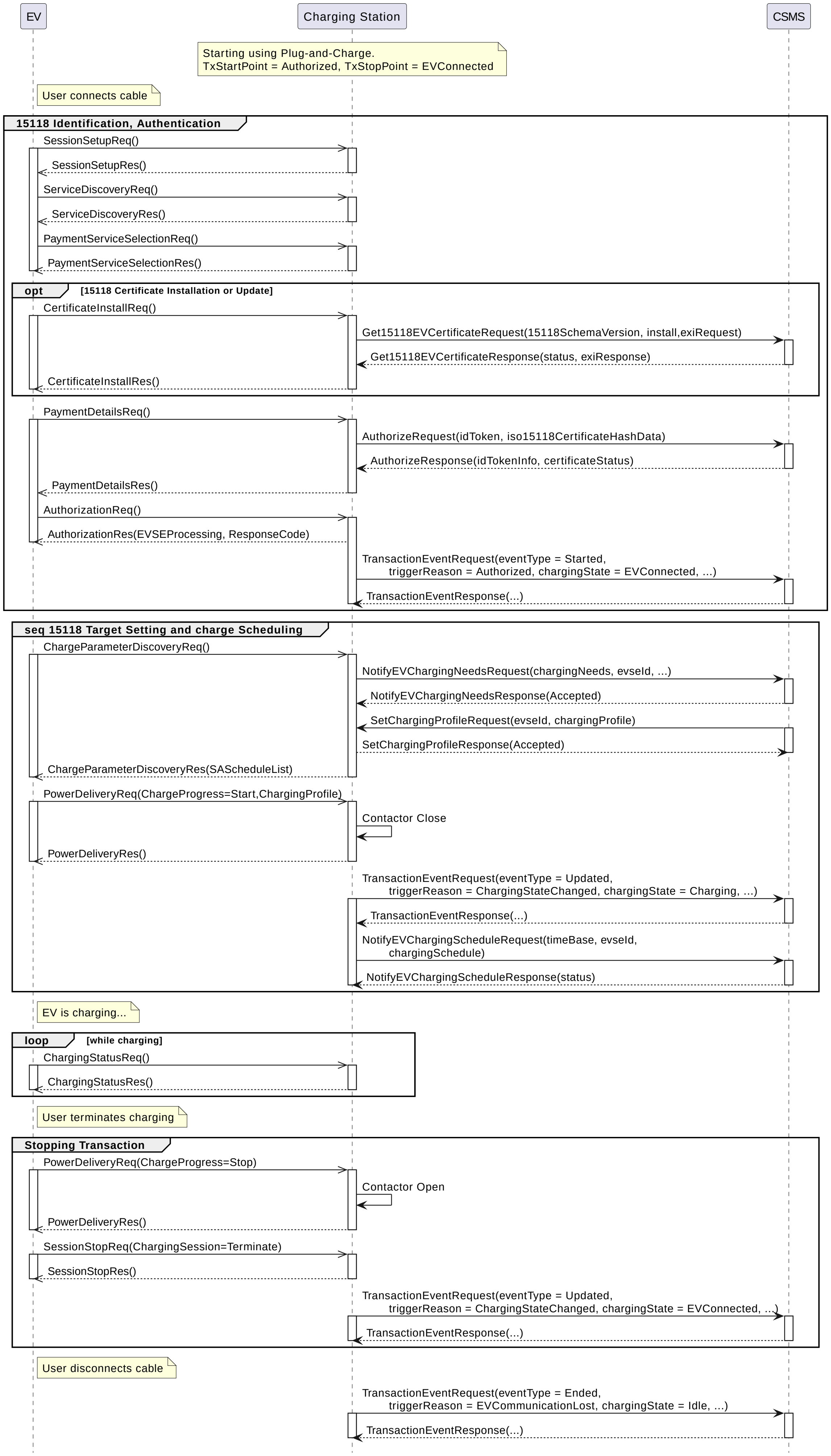
EV
Charging Station
CSMS
Starting using Plug-and-Charge.
TxStartPoint = Authorized, TxStopPoint = EVConnected
User connects cable
15118 Identification, Authentication
SessionSetupReq()
SessionSetupRes()
ServiceDiscoveryReq()
ServiceDiscoveryRes()
PaymentServiceSelectionReq()
PaymentServiceSelectionRes()
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 10. 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].

2.38. Page 330 - (2025-01) - N03 Set Monitoring Base: Improved text of Remark

Old	**8**	**Remark(s)**	Upon receipt of a SetMonitoringBaseRequest for `HardWiredOnly` or `FactoryDefault` the Charging Station will discard of any previously configured custom monitors and will activate the monitoring settings that are related to given MonitoringBase. For a MonitoringBase = `All` the Charging Station will activate all pre-configured monitors and leave previously configured custom monitors intact. This includes the custom monitors that were created when changing an existing pre-configured monitor. When the set of pre-configured monitors for `All` and `FactoryDefault` is the same, then the difference between the two is, that with `FactoryDefault` all custom monitors are deleted before the factory default pre-configured monitors are restored.
New	**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.

2.39. Page 331 - (2025-01) - N03.FR.04: text improvement

Improved requirement text for N03.FR.04.

Old	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 default monitoring settings as recommended by the manufacturer.
New	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.

2.39.1. Page 434 - MonitoringBaseEnumType description update

The description has been updated to be better aligned with the use case.

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 monitoring settings ~~has recommended by the manufacturer. This is a subset of all pre-configured monitors.~~ of Charging Station and remove all custom monitors.
HardWiredOnly	Removes all custom monitors and disables all pre-configured monitors.

2.40. Page 446 - (2025-01) - ActiveNetworkProfile is incorrectly marked as optional

The referenced variable OCPPCommCtrlr.ActiveNetworkProfile is incorrectly marked as optional. As the description states this variable needs to be implemented when the Charging Station supports the NetworkConnectionProfile use cases B09/B10. These use cases are an integral part of the Core of OCPP. In addition, all other NetworkConnectionProfile related referenced variables are already required; OCPPCommCtrlr.NetworkConfigurationPriority and OCPPCommCtrlr.NetworkProfileConnectionAttempts. Therefore, OCPPCommCtrlr.ActiveNetworkProfile should also be marked as required.

2.40.1. `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.		

2.41. Page 453 - (2025-01) - References to *monitorValue* changed to *value* [354]

The specification in N Diagnostics consistently refers to a variable *monitorValue*, but this variable is called *value* in JSON schemas.

All occurences of *monitorValue* in N Diagnostics have been changed to *value*.

2.42. Page 454 - (2025-01) - N04.FR.06 Improved limit definition of thresholds [353]

The requirement has been defined more exact, because limits for upper and lower differ slightly.

	ID	Precondition	Requirement definition	Note
Old	N04.FR.06	When the Charging Station receives a SetVariableMonitoringRequest with monitor type UpperThreshold or LowerThreshold AND the *monitorValue* is lower or higher than the range of the given Variable	The Charging Station SHALL set the attributeStatus field in the corresponding SetMonitoringResult to: Rejected.	More information can be provided in the optional *statusInfo* element.

	ID	Precondition	Requirement definition	Note
New	N04.FR.06	When the Charging Station receives a SetVariableMonitoringRequest with (monitor *type* = `UpperThreshold` AND *monitorValue* < *minLimit* OR *monitorValue* > *maxLimit*) OR (monitor *type* = `LowerThreshold` AND *monitorValue* < *minLimit* OR *monitorValue* > *maxLimit*)	The Charging Station SHALL set the attributeStatus field in the corresponding SetMonitoringResult to: Rejected.	*minLimit* and *maxLimit* refer to the [cmn_variablecharacteristicstype] for the [cmn_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.

2.43. Page 458 - (2025-01) - Added optional variable to allow the Charging Station to report its supported idTokenTypes

Currently there is no method for the Charging Station to report which idTokenTypes it supports. This would be very useful for a CSMS to know and will improve the automated onboarding of Charging Stations.

2.43.1. `SupportedIdTokentypes`

Required	no		
Component	**componentName**	AuthCtrlr	
Variable	**variableName**	SupportedIdTokenTypes	
	variableAttributes	**mutability**	ReadOnly
	variableCharacteristics	**dataType**	MemberList
		valuesList	List of IdTokenEnumType.
Description	The subset of the list of supported IdTokenTypes as defined by IdTokenEnumtype, that is supported by the Charging Station.		

2.44. Page 462 - (2025-01) - Added note to EnergyTransfer description as TxStartPoint

A note has been added to the description of the EnergyTransfer TxStartPoint to warn for potential skews of the values of the energy meter readings associated with start of the transaction.

Value	Description
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.

2.45. Page 476 - (2025-01) - Added Connector component to AvailabilityState referenced variable

It was already possible to report the AvailabilityState of the Connector component, however the definition was missing at this table.

2.45.1. 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.		

3. Part 3

Currently no new errata for OCPP 2.0.1 part 3.

4. Part 4

4.1. Page 6 - (2025-01) - 3.1.1. The connection URL

The following clarifying text was added:

Old text	[...] percent-encoded [...]
New text	[...] percent-encoded / URL encoded [...]

4.2. Page 7 - (2025-01) - 3.1.2. OCPP version

The following text has been rewritten for clarity:

Old text	The OCPP version should not be part of the OCPP-J endpoint URL string if you want to select the OCPP version to use via the websocket protocol negotiation mechanism, as explained in Server Response.
New text	If the OCPP version is part of the OCPP-J endpoint URL it SHALL not determine the OCPP version to use, because the OCPP version is selected via the websocket protocol negotiation mechanism, as explained in section 3.3 [server-response].

4.3. Page 8 - (2025-01) - 3.3. WebSocket Compression

The following text was duplicate. The duplication has been removed and the text has been clarified:

Old text	OCPP Requires the CSMS (and Local Controller) to support RFC 7692, WebSocket compression is seen as a relative simple way to reduce mobile data usage. For a Charging Station this is not a hard requirement, as this might be more complex to implement on an embedded platform, but as this is seen as efficient solution to reduce mobile data usage, it is RECOMMENDED to be implemented on a Charging Station that uses a mobile data connection. OCPP Requires the CSMS (and Local Controller) to support RFC 7692, WebSocket compression is seen as a relative simple way to reduce mobile data usage. For a Charging Station this is not a hard requirement, as this might be more complex to implement on an embedded platform, but as this is seen as efficient solution to reduce mobile data usage, it is RECOMMENDED to be implemented on a Charging Station that uses a mobile data connection.
New text	The CSMS (and Local Controller) SHALL support RFC 7692, WebSocket compression, which is a relative simple way to reduce mobile data usage. For a Charging Station this is not a hard requirement, as this might be more complex to implement on an embedded platform. It is RECOMMENDED to be implemented on a Charging Station, because it is an efficient solution to reduce mobile data usage.

4.4. Page 9 - (2025-01) - 4.1.1. Synchronicity

The following Note was added:

NOTE A Charging Station or CSMS is allowed to send a SEND message to the other party before the CALL message it has sent has been responded to or has timed out.

4.5. Page 10 - (2025-01) - 4.1.3. The message type

OLD: |CALL |2 | Request message |CALLRESULT |3 | Response message

New:

|CALL |2 | Request message, i.e. messages ending in "Request" |CALLRESULT |3 | Response message, i.e. message ending in "Response"

4.6. Page 10 - (2025-01) - 4.1.3. The message type

Refer to 'system', instead of 'server'.

Old text	When a server receives a message with a Message Type Number not in this list, it SHALL ignore the message payload. Each message type may have additional required fields.
New text	When a system receives a message with a Message Type Number not in this list, it SHALL ignore the message payload. Each message type may have additional required fields.

4.7. Page 10 - (2024-09) - 4.1.4. Message ID

Change the following text in paragraph 4.1.4:

Old text	The message ID serves to identify a request. A message ID for any CALL message MUST be different from all message IDs previously used by the same sender for any other CALL messages on any WebSocket connection using the same unique Charging Station identifier. ~~This also applies to retries of messages.~~
New text	The message ID serves to identify a request. A message ID for any CALL message MUST be different from all message IDs previously used by the same sender for any other CALL messages on any WebSocket connection using the same unique Charging Station identifier. The message ID for a retried message (e.g. when no response was received within timeout) MAY be identical to the message ID of the original message.

4.8. Page 13 - (2025-01) - 4.2.3. CALLERROR

In the example, the ErrorDescription text has been updated:

Old text	"SetDisplayMessageRequest not implemented"
New text	"SetDisplayMessageRequest not supported"

4.9. Page 15 - (2025-01) - 5. Connection

The following clarifying text has been added related to the TLS fragment length:

4.10. TLS fragment length

TLS involves sending "Records" between peers. Records can be of type “Handshake”, “Alert”, “ChangeCipherSpec”, “Heartbeat” or “Application”. OCPP messages are sent in Application records. The payload contains a “fragment” of the application data. The record layer fragments information blocks into TLSPlaintext records carrying data in chunks of 2^14 bytes (16kB) or less.

TLS peers need to maintain an input and an output buffer to store an entire fragment of 16 kB. For a low resource device it is a large cost to allocate 32 kB for the TLS connection.

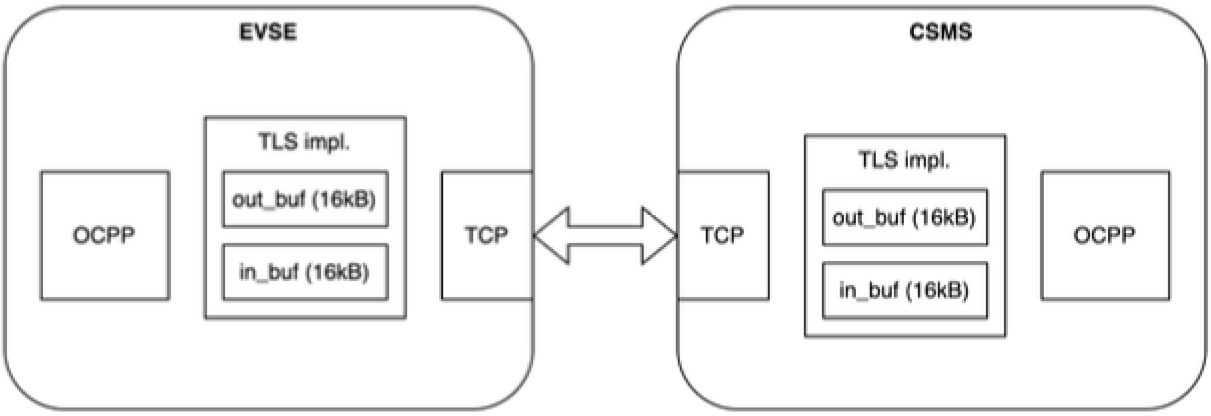

Figure 11. Peers allocating standard 16 kB TLS buffers

A TLS extension is defined in TLS Extensions RFC6066 Section 4, that allows the client to ask for a different maximum fragment length than the default 16kB. A client can ask for a maximum fragment length of 0.5 kB, 1 kB, 2 kB or 4 kB. This TLS extension is, however, not widely supported and native managed cloud TLS termination services typically don’t support this.

A resource-constrained Charging Station SHOULD try to negotiate a smaller TLS maximum fragment size, and if that is not accepted by the peer, then Charging Station MAY unilaterally decide to allocate less memory to its TLS output buffer. A TLS maximum fragment length of 2 kB is suggested based on data collection during certification tests, which shows that 99% of the messages fit in a 2 kB buffer.

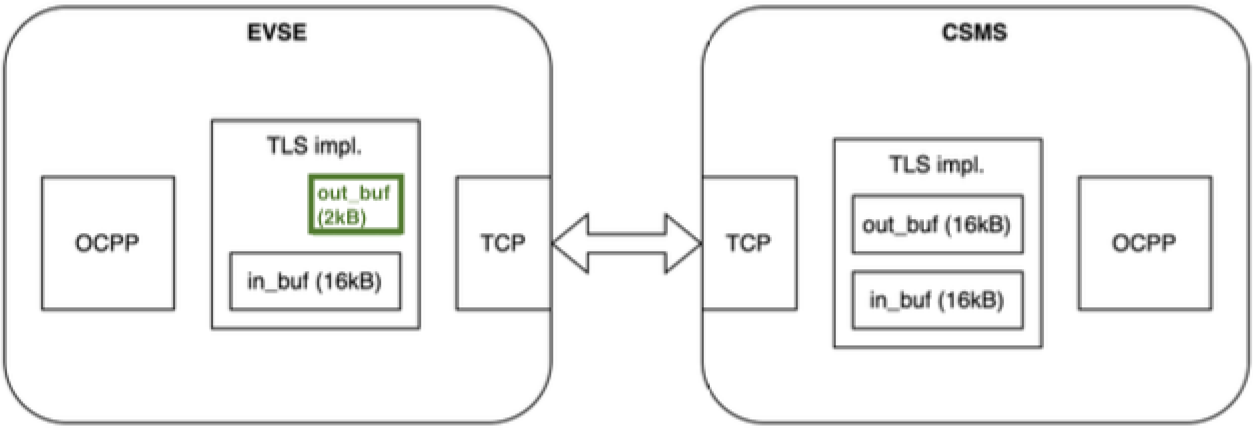

Figure 12. Charging Station allocating a 2 kB TLS output buffer

4.11. Page 15 - (2025-01) - 5.3. WebSocket Ping in relation to OCPP Heartbeat

The following clarifying text was added:

New text	A Heartbeat message checks connectivity end-to-end, whereas a Websocket ping/pong only checks from point-to-point. This makes a difference in an extended network topology with a Local Controller between Charging Station and CSMS.

4.12. Page 15 - (2025-01) - Chapter 5 Connection - Added section about TLS fragment length

As a result of the discussions at the OCPP lite taskgroup, a section has been added providing guidance on implementing TLS fragment negotiation.

5. Part 5

5.1. Page 9 - (2024-11) - Optional feature list for charging station - Change name R-3

The specified name of feature R-3 is not correct. It should reflect the ability to disable reservations.

Old	R-3	Reservation support (ReservationEnabled)	Configuration Variable for H01
New	R-3	Support for disabling Reservations (ReservationEnabled)	Configuration Variable for H01

5.2. Page 19 - (2024-09) - TC_E_04_CS Updated condition for test case to exclude it for MacAddress and ISO 15118 PnC

This test case cannot be performed with the local authorization option MacAddress or ISO 15118 PnC.

		Local start transaction - Authorization first					
Old	TC_E_04	Success	C	M	Applicable if one or more of the local start authorization options is implemented.	C-30 - C-35 or ISO 15118 support	Authorization options for local start
New	TC_E_04	Success	C	M	Applicable if one or more of the local start authorization options is implemented.	(C-30 or C-31 or C-32 or C-33 or C-35)	Authorization options for local start

5.3. Page 22 - (2024-09) - TC_E_17_CS Updated condition for test case to correctly specify the applicable TxStopPoint combinations

This testcase allows for a limited set of TxStopPoint combinations, otherwise it is not applicable.

		Local start transaction - Authorization first					
Old	TC_E_17	Deauthorized - EV side disconnect	C	M	- TxStopPoint can either be ReadOnly with a subset of the values or have a valueList of supported values, that contains a subset. This testcase is applicable if the value Authorized or PowerPathClosed is a supported value. - StopTxOnEVSideDisconnect needs to ReadWrite or ReadOnly with value true	(C-10.2 or C-10.3) and C-06.2 and AQ-9	Supported Transaction Stop points

		Local start transaction - Authorization first					
New	TC_E_17	Deauthorized - EV side disconnect	C	M	This testcase is applicable if the value Authorized is a supported value for TxStopPoint AND EVConnected, PowerPathClosed and EnergyTransfer must not be set as TxStopPoint AND StopTxOnEVSideDisconnect true must be a supported value.	C-10.2 and C-06.2 and AQ-9 and NOT (NOT C-52 AND (10.1 OR C-10.3 OR 10.4))	Supported Transaction Stop points

5.4. Page 40 - (2024-11) - TC_H_13_CS Updated invalid condition for test case

The condition should have been reversed.

Old	TC_H_13	Rejected	C		Depending on configuration variable ReservationNonSpecificEVSE	R-2	Support reservations of unspecified EVSE
New	TC_H_13	Rejected	C		Depending on the Charging Station not supporting the configuration variable ReservationNonSpecificEVSE	NOT R-2	Support reservations of unspecified EVSE

5.5. Page 48 - (2024-06) - Added additional questions to appendix

The following additional questions are added for CSMSs:

Id	**Additional questions for lab testing**
AQ-3	Does your CSMS support Absolute values for the following Charging Profiles:
AQ-3.1	*TxDefaultProfile*
AQ-3.2	*ChargingStationMaxProfile*
AQ-4	Does your CSMS support Recurring values for the following Charging Profiles:
AQ-4.1	*TxDefaultProfile*
AQ-4.2	*ChargingStationMaxProfile*

6. Part 6

6.1. General

6.1.1. Page XX - (2024-11) - All testcases - Updated table structure of all testcases

The table structure of all testcases have been updated. This has been done for multiple reasons:

- It improves readability by providing more space for the main steps.
- It decreases the chance of testcase tables being broken, resulting in missing steps at the bottom of a testcase.
- It makes it easier for the Technical Editors to update testcases.

6.2. Charging Station

6.2.1. Page 7 - (2024-11) - TC_A_05_CS - Successfully reconnecting after every failed connection attempt

It is needed to reconnect after every (intended) failed connection, otherwise the retryBackoffTime may double itself several time, resulting in a very large number that may exceed the configured timeouts.

Table 5. Test Case Id: TC_A_05_CS

Test case name	**TLS - server-side certificate - Invalid certificate**
Test case Id	TC_A_05_CS
Use case Id(s)	A00
Requirement(s)	A00.FR.309,A00.FR.310,A00.FR.311,A00.FR.412,A00.FR.413,A00.FR.414
System under test	Charging Station
Description	The CSMS uses a server-side certificate to identify itself to the Charging Station, when using security profile 2 or 3.
Purpose	To verify whether the Charging Station is able to terminate the connection when the received server certificate is invalid.
Prerequisite(s)	- The charging station supports security profile 2 and/or 3 - The active NetworkConnectionProfile uses either security profile 2 OR 3. *<Removed>*
Before (Preparations)	**Configuration State:** **OCPPCommCtrlr.NetworkConfigurationPriority** only contains *<Value from ActiveNetworkProfile>*
	Memory State: N/a
	Reusable State(s): N/a

Test case name	**TLS - server-side certificate - Invalid certificate**	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The OCTT aborts the connection with the Charging Station.	
	2. The Charging Station initiates a TLS handshake and sends a Client Hello to the OCTT.	**3.** The OCTT responds with a Server Hello With a <Configured **valid** server certificate> Note(s): *- The OCTT will use this as an indication of the time it takes the Charging Station to reconnect.*
	4. The OCTT aborts the connection with the Charging Station.	
	5. The Charging Station initiates a TLS handshake and sends a Client Hello to the OCTT.	**6.** The OCTT responds with a Server Hello With a <Configured **invalid** server certificate>
	7. The Charging Station deems the server certificate invalid and terminates the connection.	
	Note: *The OCTT will wait two times the measured reconnection time from step 3, before switching the server certificate back to the valid server certificate. The reason for this is that the OCTT is not always able to detect a failed connection attempt.*	
	8. The Charging Station initiates a TLS handshake and sends a Client Hello to the OCTT.	**9.** The OCTT responds with a Server Hello With a <Configured **valid** server certificate> Note(s): *- The OCTT will accept the connection to prevent doubling of the RetryBackOffWaitMinimum.*
	10 The Charging Station sends a **SecurityEventNotificationRequest**	**11** The OCTT responds with a **SecurityEventNotificationResponse**
	Note(s): *- Steps 4 to 11 are repeated per configured invalid server certificate.* *- In case default certificates are being used, the OCTT will use three different invalid server certificates; "Not signed by installed Root certificate", "Expired", "CommonName that does not equal the FQDN of the server".* *- In case custom certificates are being used, the OCTT will loop through all certificates configured at the 'CSMS Keystore Invalid'.*	
Tool validations	* Step 11: Message: **SecurityEventNotificationRequest** - **type** must be *InvalidCsmsCertificate*	
	Post scenario validations: N/a	

6.2.2. Page 22 - (2024-09) - TC_A_19_CS - Fixed references to ConfigurationSlot [O20-4762]

Test case name	**Upgrade Charging Station Security Profile - Accepted**
Test case Id	TC_A_19_CS

Test case name	**Upgrade Charging Station Security Profile - Accepted**	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SetNetworkProfileResponse**	**1.** The OCTT sends a **SetNetworkProfileRequest** with **configurationSlot** is *<Configured configurationSlot>* or *<Configured configurationSlot2>* depending on which one is already in use - **connectionData.messageTimeout** *<Configured messageTimeout>* - **connectionData.ocppCsmsUrl** *<Configured ocppCsmsUrl>* - **connectionData.ocppInterface** *<Configured ocppInterface>* - **connectionData.ocppVersion** *OCPP20* - **connectionData.securityProfile** *<Configured securityProfile + 1>*
	4. The Charging Station responds with a **SetVariablesResponse**	**3.** The OCTT sends a **SetVariablesRequest** with **variable.name** is *"NetworkConfigurationPriority"* **component.name** is *"OCPPCommCtrlr"* **attributeValue** is *"<Configured configurationSlot2>,<Configured configurationSlot>"*
	...	...
Tool validations	* Step 2: Message **SetNetworkProfileResponse** - **status** *Accepted* * Step 4: Message **SetVariablesResponse** - **setVariableResult[0].attributeStatus** *Accepted* OR *RebootRequired* * Step 6: Message **ResetResponse** - **status** *Accepted* * Step 11: Message **GetVariablesResponse** - **getVariableResult[0].attributeValue** *<Configured securityProfile + 1>* * Step 13: Message **GetVariablesResponse** - **getVariableResult[0].attributeValue** Does not contain the configurationSlot with the previous (lower) security profile	
	Post scenario validations: - N/a	

6.2.3. Page 56 - (2024-09) - TC_B_20_CS - Added check on omitting evseId [4390]

Test case name	**Reset Charging Station - Without ongoing transaction - OnIdle**	
Test case Id	TC_B_20_CSMS	
...		
Main (Test scenario)	**Charging Station**	**CSMS**
	...	
Tool validations	* Step 1: Message **ResetRequest** - **evseId** must be omitted * Step 4: Message **BootNotificationResponse** - **status** *Accepted*	
	Post scenario validations: - N/a	

6.2.4. Page 57 - (2024-09) - TC_B_21_CS - Added check on omitting evseId [4390]

Test case name	**Reset Charging Station - With Ongoing Transaction - OnIdle**	
Test case Id	TC_B_21_CSMS	
…		
Main (Test scenario)	**Charging Station**	**CSMS**
	…	
Tool validations	* Step 1: Message **ResetRequest** - **type** *OnIdle* - **evseId** must be omitted * Step 8: Message **BootNotificationResponse** - **status** *Accepted*	
	Post scenario validations: - N/a	

6.2.5. Page 58 - (2024-09) - TC_B_22_CS - Added check on omitting evseId [4390]

Test case name	**Reset Charging Station - With Ongoing Transaction - Immediate**	
Test case Id	TC_B_22_CSMS	
…		
Main (Test scenario)	**Charging Station**	**CSMS**
	…	
Tool validations	* Step 1: Message **ResetRequest** - **type** *Immediate* - **evseId** is omitted * Step 6: Message **BootNotificationResponse** - **status** *Accepted*	
	Post scenario validations: - N/a	

6.2.6. Page 138 - (2024-09) - TC_C_47_CS - StoppedReason must be validated in Ended event [O20-4467]

Test case name	**Stop Transaction with a Master Pass - With UI - All transactions**	
Test case Id	TC_C_47_CS	
…		
Main (Test scenario)	**Charging Station**	**CSMS**
	…	…

Test case name	Stop Transaction with a Master Pass - With UI - All transactions
Tool validations	* Step 1: Message **AuthorizeRequest** - **idToken.idToken** *<Configured masterpass_idtoken_idtoken>* - **idToken.type** *<Configured masterpass_idtoken_type>* * Step 3: Message **TransactionEventRequest** - **transactionInfo.stoppedReason** *MasterPass* (in last TransactionEventRequest) - **idToken** *omit* or - **idToken.idToken** *<Configured masterpass_idtoken_idtoken>* and - **idToken.type** *<Configured masterpass_idtoken_type>* (once per stopped transaction) - **eventType** *Ended* (in last TransactionEventRequest)
	Post scenario validations: - N/a

6.2.7. Page 146 - (2024-11) - TC_C_26_CS - Allow StatusNotification status = Occupied

The connector status change should be reported after the connection is restored. The Charging Station should be allowed to report StatusNotificationRequest status = Occupied (or NotifyEventRequest).

Test case name	Set Variable Monitoring - Periodic event	
Test case Id	TC_C_26_CS	
Use case Id(s)	C15 & C13	
Requirement(s)	C15.FR.02,C15.FR.06,C15.FR.08,C13.FR.04	
…		
Main (Test scenario)	**Charging Station**	**CSMS**
	…	…
	1. The Charging Station notifies the CSMS about the current state of all connectors.	**2.** The OCTT responds accordingly.
	…	…
	3. Execute **Reusable State** *StopAuthorized*	
	4. Execute **Reusable State** *EVConnectedPostSession*	
	5. Execute **Reusable State** *EVDisconnected*	
Tool validations	* Step 1: Message: **StatusNotificationRequest** - **connectorStatus** must be *Occupied* Message: **NotifyEventRequest** - **eventData[0].trigger** must be *Delta* - **eventData[0].actualValue** must be *Occupied* - **eventData[0].component.name** must be *Connector* - **eventData[0].variable.name** must be *AvailabilityState* …	
	Post scenario validations: N/A	

6.2.8. Page 151 - (2024-09) - TC_C_54_CS - removed reusable state IdTokenCached [O20-3510]

Test case name	Authorization using Contract Certificates 15118 - Offline - ContractValidationOffline is true
Test case Id	TC_C_54_CS
…	

Test case name	Authorization using Contract Certificates 15118 - Offline - ContractValidationOffline is true	
Before (Preparations)	**Configuration State:** ...	
	Memory State: *CertificateInstalled* for certificateType *V2GRootCertificate* *CertificateInstalled* for certificateType *MORootCertificate* ~~*IdTokenCached* for <*Configured valid IdToken fields*> (If implemented)~~ *IdTokenLocalAuthList* for <*Configured valid IdToken fields*> (If implemented)	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	...	...
...		

6.2.9. Page 153 - (2024-09) - TC_C_55_CS - removed reusable state IdTokenCached [O20-3510]

Test case name	Authorization using Contract Certificates 15118 - Offline - ContractValidationOffline is false	
Test case Id	TC_C_55_CS	
...		
Before (Preparations)	**Configuration State:** ...	
	Memory State: *CertificateInstalled* for certificateType *V2GRootCertificate* *CertificateInstalled* for certificateType *MORootCertificate* ~~*IdTokenCached* for <*Configured valid IdToken fields*> (If implemented)~~ *IdTokenLocalAuthList* for <*Configured valid IdToken fields*> (If implemented)	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	...	...
...		

6.2.10. Page 174 - (2024-09) - TC_E_17_CS -Updated prerequisite for test case to correctly specify the applicable TxStopPoint combinations

This testcase allows for a limited set of TxStopPoint combinations, otherwise it is not applicable.

Test case name	Stop transaction options - Deauthorized - EV side disconnect
Test case Id	TC_E_17_CS
...	
Old: Prerequisite(s)	- The Charging Station does NOT have the following configuration; The mutability of **TxStopPoint** is *ReadOnly* AND the value *Authorized* OR *PowerPathClosed* is NOT set OR (*EnergyTransfer* OR *DataSigned* OR *EVConnected is set).* - *If the mutability of* ***TxStopPoint*** *is _ReadWrite*, then the value *Authorized* OR *PowerPathClosed* must be supported.
New: Prerequisite(s)	This testcase is applicable if the value Authorized is a supported value for TxStopPoint AND EVConnected, PowerPathClosed and EnergyTransfer must not be set as TxStopPoint AND StopTxOnEVSideDisconnect true must be a supported value.

6.2.11. Page 176 - (2024-11) - TC_E_39_CS - Missing StatusNotificationRequest/NotifyEventRequest

Test case name	**Stop transaction options - Deauthorized - timeout**	
Test case Id	TC_E_39_CS	
...		
Main (Test scenario)	**Charging Station**	**CSMS**
	...	...
	Manual Action: *Connect the EV and EVSE on EV side.*	
	Manual Action: *Connect the EV and EVSE on EVSE side.*	
	3. The Charging Station notifies the CSMS about the status change of the connector.	**4.** The OCTT responds accordingly.
	...	...
Tool validations	* Step 1: Message: **TransactionEventRequest** - **triggerReason** must be *EVConnectTimeout* - **eventType** must be *Updated* if TxStartPoint is *ParkingBayOccupancy*, else *Ended* - **transactionInfo.stoppedReason** must be *Timeout* * Step 3: Message: **StatusNotificationRequest** - **connectorStatus** must be *Occupied* Message: **NotifyEventRequest** - **eventData[0].trigger** must be *Delta* - **eventData[0].actualValue** must be *Occupied* - **eventData[0].component.name** must be *Connector* - **eventData[0].variable.name** must be *AvailabilityState* * Step 5: Message: **TransactionEventRequest** - **triggerReason** can only be *CablePluggedIn* - **transactionInfo.chagringState** should not be *Charging* - **eventType** must be *Updated* if TxStartPoint is *ParkingBayOccupancy*, else *Ended*	
	Post scenario validations: N/a	

6.2.12. Page 185 - (2024-09) - TC_E_35_CS - StoppedReason must be validated in Ended event [O20-4467]

Test case name	**Stop transaction options - PowerPathClosed - Remote stop**	
Test case Id	TC_E_35_CS	
...		
Main (Test scenario)	**Charging Station**	**CSMS**
	...	...
Tool validations	* Step 2: Message: **RequestStopTransactionResponse** - **status** must be *Accepted* * Step 3: Message: **TransactionEventRequest** - **triggerReason** must be *RemoteStop* (for one of the TransactionEventRequests) - **transactionInfo.stoppedReason** must be *Remote* (for the last TransactionEventRequest) - **eventType** must be *Ended* (for the last TransactionEventRequest)	
	Post scenario validations: N/a	

6.2.13. Page 214 - (2024-06) TC_E_43_CS Move reusable state TransactionEventsInQueueEnded to Before [768]

State TransactionEventsInQueueEnded is moved to Before stage.

Test Case Id: TC_E_43_CS

Test case name	**Offline Behaviour - Transaction during offline period**	
Test case Id	TC_E_43_CS	
Use case Id(s)	E12	
Requirement(s)	E12.FR.01,E12.FR.02,E12.FR.06	
System under test	Charging Station	
Description	The Charging Station queues TransactionEvent messages to inform the CSMS that a transaction occurred while the Charging Station was Offline.	
Purpose	To verify if the Charging Station is able to queue TransactionEvent messages while it was offline.	
Prerequisite(s)	The Charging Station supports authorization methods other than NoAuthorization	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): **State is** *TransactionEventsInQueueEnded*	
Main (Test scenario)	**Charging Station**	**CSMS**
	~~**1.** Execute **Reusable State** *TransactionEventsInQueueEnded*~~	
	1. The Charging Stations sends a **TransactionEventRequest** Note(s): *- The Charging Station will empty its Transaction message queue. This will contain one or more TransactionEventRequest messages*	**2.** The OCTT responds with a **TransactionEventResponse**
Tool validations	* Step 1: All messages: **TransactionEventRequest** - **offline** must be *true* One of the messages: **TransactionEventRequest** - **eventType** *Started* One of the messages: **TransactionEventRequest** - **eventType** *Ended*	
	Post scenario validations: N/a	

NOTE If the Charging Station supports ISO15118, this testcase needs to be executed using EIM.

6.2.14. Page 221 - (2024-06) TC_F_04_CS Made mandatory in part 5, but prerequisite in part 6 was not updated

Removed Prerequisite(s):

Old	The Charging Station supports TxCtrlr.TxStartPoint ParkingBayOccupancy OR Authorized.
New	N/a

6.2.15. Page 295 - (2024-09) - TC_J_XX_CS Meter Values

Meter values cannot have *location* = "EV", unless it is for measurand "SoC".

For all test cases in J add the following Post scenario validation:

...	
Tool validations	...
	Post scenario validations: Message: **MeterValuesRequest/TransactionEventRequest** - ... - None of the provided **sampledValue** s shall have **location** = *EV* , except when **measurand** = *SoC* .

6.2.16. Page 297 - (2024-09) - TC_J_02_CS Clock-aligned Meter Values

Meter values must be reported for all phases.

Test case name	**Clock-aligned Meter Values - Transaction ongoing**	
Test case Id	TC_J_02_CS	
Use case Id(s)	J01	
...		
Before (Preparations)	**Configuration State:** **AlignedDataInterval** is *<Configured clock-aligned Meter Values interval>* **AlignedDataSendDuringIdle** is *false* (If implemented) **RegisterValuesWithoutPhases** is *false* (If implemented)	
	Memory State: N/a	
	Reusable State(s): **State is** *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	...	
Tool validations	*Note: The following steps do not need to be sent in a specific order.* * Step 1: Message: **MeterValuesRequest** - **meterValue[0].sampledValue[0].context** must be *Sample.Clock* - **meterValue[0].sampledValue** must contain *<An element per configured measurand at the AlignedDataMeasurands for the number of phases reported in* ***SupplyPhases****. The measurand field may be omitted when the measurand is "Energy.Active.Import.Register">* * Step 3: Message: **TransactionEventRequest** - **triggerReason** must be *MeterValueClock* - **metervalue[0].sampledValue[0].context** must be *Sample.Clock* - **metervalue[0].sampledValue** must contain *<An element per configured measurand at the AlignedDataMeasurands for the number of phases reported in* ***SupplyPhases****. The measurand field may be omitted when the measurand is "Energy.Active.Import.Register">*	
	Post scenario validations: ...	

6.2.17. Page 306 - (2024-06) - TC_J_10_CS - Remove reference to non-existing requirements [4697]

Test case name	**Sampled Meter Values - EventType Ended**
Test case Id	TC_J_10_CS
Use case Id(s)	J02 & (E06,E07,E08,E09,E10,E12)

Test case name	**Sampled Meter Values - EventType Ended**
Requirement(s)	J02.FR.01,J02.FR.02,J02.FR.03,J02.FR.04,J02.FR.10, E06.FR.11, ~~E06.FR.17~~, E07.FR.08, ~~E07.FR.13~~,E08.FR.09,E09.FR.05,E10.FR.04,E12.FR.07
…	

6.2.18. Page 345 - (2024-06) TC_K_35_CS Get Charging Profile - EvseId > 0 + chargingProfilePurpose [773]

Change initial charging state from "N/A" to:

	Charging State: **State is** *EnergyTransferStarted*

6.2.19. Page 362 - (2024-09) - TC_K_56_CS Removed expecting triggerReason=ChargingRateChanged [776]

A trigger reason ChargingStateChange must only be sent, when an external actor (not CSMS) changes the charging rate. Therefore, removed the check that triggerReason=ChargingStateChanged is sent. Also added a check that the EV charging schedule fits within the given charging profile.

Test case name	**Charging with load leveling based on High Level Communication - Offline**	
Test case Id	TC_K_56_CS	
…		
Main (Test scenario)	**Charging Station**	**CSMS 6.** The OCTT responds with a **TransactionEventResponse**.
	…	
Tool validations	* Step 3: (Message: **NotifyEVChargingScheduleRequest**) **evseId** *<Configured evseId>* **chargingSchedule.chargingSchedule[0].chargingRateUnit** *<Configured chargingRateUnit>* **chargingSchedule.chargingSchedule[0].chargingSchedulePeriod[0].startPeriod** *0* If <Configured chargingRateUnit> is W: **chargingSchedule.chargingSchedule[0].chargingSchedulePeriod[0].limit** <= *8000* Else: **chargingSchedule.chargingSchedule[0].chargingSchedulePeriod[0].limit** <= *8* * ~~Step 5:~~ ~~Message: **TransactionEventRequest**~~ ~~- **triggerReason** must be *ChargingStateChanged*~~ ~~- **transactionInfo.chargingState** must be *Charging*~~ ~~- **offline** *true*~~	
	Post scenario validations: N/a	

6.2.20. Page 468 - (2024-06) TC_N_23_CS Offline Notification - OfflineMonitoringEventQueuingSeverity set higher than severityLevel of the monitor [772]

Test case name	**Offline Notification - OfflineMonitoringEventQueuingSeverity set higher than severityLevel of the monitor**
Test case Id	TC_N_23_CS
Use case Id(s)	N07
Requirement(s)	N07.FR.04
System under test	Charging Station
Description	Charging Station does not queue event notifications when offline.
Purpose	To test that Charging Station does not queue event notifications with a severity higher than `OfflineMonitoringEventQueuingSeverity`.

Test case name	**Offline Notification - OfflineMonitoringEventQueuingSeverity set higher than severityLevel of the monitor**	
Prerequisite(s)	Charging Station is online at start of test for configuration. CS has implemented device model monitoring and MonitoringCtrlr::Enabled = true.	
Before (Preparations)	**Configuration State:** SetConfiguration with: - **component.name** = *"MonitoringCtrlr"* - **variable.name** = *"OfflineQueuingSeverity"* - **attributeValue** = *<Configured Severity>*	
	Memory State: Charging Station has custom or predefined monitors on variable AvailabilityState of Configured EVSE and Configured ConnectorId with *severity = <Configured severity> + 1*	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Connect the EV and EVSE.*	
	1. The Charging Station notifies the CSMS about the status change of the connector.	**2.** The OCTT responds accordingly.
	Note(s): *Step 3, 4, 5, 6, 7, and 8 need to be executed when **TxStartPoint** contains EVConnected OR ParkingBayOccupancy*	
	3. The Charging Station sends a **TransactionEventRequest**	**4.** The OCTT responds with a **TransactionEventResponse**
	Manual Action: *Take Charging Station offline.*	
	Manual Action: *Disconnect the EV and EVSE.*	
	Manual Action: *Connect the EV and EVSE.*	
	Note(s): *The tool will now wait for <Configured Transaction Duration> seconds*	
	Manual Action: *Bring Charging Station back online.*	
	5. The Charging Station sends a **TransactionEventRequest**	**6.** The OCTT responds with a **TransactionEventResponse**
	7. The Charging Station sends a **TransactionEventRequest**	**8.** The OCTT responds with a **TransactionEventResponse**
	Note(s): *The CS shall not send a NotifyEventRequest for AvailabilityState of EVSE and Connector. A StatusNotification may still be received.*	

Test case name	**Offline Notification - OfflineMonitoringEventQueuingSeverity set higher than severityLevel of the monitor**
Tool validations	* Step 1: *(Optional:)* Message: **StatusNotificationRequest** - **evseId** *<configured evseId>* - **connectorId** *<configured connectorId>* - **connectorStatus** must be *Occupied* *(Required, but can be combined into one NotifyEventRequest:)* Message: **NotifyEventRequest** - **eventData[0].trigger** must be *Delta* - **eventData[0].actualValue** must be *Occupied* - **eventData[0].component.name** must be *Connector* - **eventData[0].component.evse.id** must be *Configured EVSE* - **eventData[0].component.evse.connectorId** must be *Configured ConnectorId* - **eventData[0].variable.name** must be *AvailabilityState* Message: **NotifyEventRequest** - **eventData[0].trigger** must be *Delta* - **eventData[0].actualValue** must be *Occupied* - **eventData[0].component.name** must be *EVSE* - **eventData[0].component.evse.id** must be *Configured EVSE* - **eventData[0].variable.name** must be *AvailabilityState*
	* Step 3: Message: **TransactionEventRequest** - **triggerReason** must be *CablePluggedIn* - **transactionInfo.chargingState** must be *EVConnected*
	* Step 5: Message: **TransactionEventRequest** - **triggerReason** must be *EVCommunicationLost* - **transactionInfo.chargingState** must be *Idle*
	* Step 7: Message: **TransactionEventRequest** - **triggerReason** must be *CablePluggedIn* - **transactionInfo.chargingState** must be *EVConnected*
	Post scenario validations: N/A

6.2.21. Page 470 - (2024-09) - TC_N_24_CS - Referring to incorrect use case and requirements [O20-4793]

Test case name	**Set Variable Monitoring - Periodic event**	
Test case Id	TC_N_24_CS	
Use case Id(s)	N04, N08	
Requirement(s)	N04.FR.01, N04.FR.08, N08.FR.05 and N08.FR.06	
…		
Main (Test scenario)	**Charging Station**	**CSMS**
	…	…
Tool validations	…	
	Post scenario validations: N/A	

6.2.22. Page 472 - (2024-12) - TC_N_26_CS - Made test case more explicit and more time before ending

TC_N_26_CS: Retrieve Log Information - Diagnostics Log - Upload failed

Main (Test scenario)	
Charging Station	**CSMS**
2. The Charging Station responds with a **GetLogResponse**	**1.** The Test System sends a **GetLogRequest** with - **logType** *DiagnosticsLog* - **retries** *3* - **retryInterval** <Configured retryInterval> - **log.remoteLocation** *<Configured log location with non-existing path>*
…	
Note(s): ~~*- Steps 3 & 4 are optional after the first attempt.*~~ ~~*- The Charging Station will perform step (3,) 5, four times with <Configured retryInterval> seconds in between.*~~ *- Step 3-4, 5-6 and 3-6 may repeat multiple times depending on Charging Station’s implementation.* *- The Test System waits at least (3 * <Configured retryInterval>), before ending the testcase.*	

Tool validations
… * Step 3: Must be sent exactly 1 or 4 times Message **LogStatusNotificationRequest** - **status** *Uploading* - **requestId** *Same Id as the GetLogRequest* * Step 5: Must be sent exactly 1 or 4 times Message **LogStatusNotificationRequest** …
Post scenario validations: - N/a

6.2.23. Page 470 - (2024-12) - TC_N_24_CS - Test case now searches suitable variable to do test with

TC_N_24_CS: Set Variable Monitoring - Periodic event

Test case name	**Set Variable Monitoring - Periodic event**
Test case Id	TC_N_24_CS
…	
Prerequisite(s)	Charging Station has implemented device model monitoring and MonitoringCtrlr::Enabled = true. This test case assumes the device model exposes at least one component.variable which can be monitored.

Before (Preparations)
…
Reusable State(s): ~~N/a~~ **State is** *[csCommunicatedBaseReport]*

Main (Test scenario)	
Charging Station	**CSMS**
Search **[csCommunicatedBaseReport]** **.baseReportData** to get a **baseReportData.reportData** WHERE **reportData.variableCharacteristics.supportsMonitoring** is *true* AND **reportData.component.instance** is *<omitted>* AND **reportData.instance.instance** is *<omitted>* AS *<componentVariable>*	
Set the monitor to generate a periodic event notification	
2. Charging Station responds with **SetVariableMonitoringResponse**	**1.** Test System sends **SetVariableMonitoringRequest** with: ~~- **setMonitoringData[0].value** = *<Configured Clock Aligned MeterValues Interval>*~~ - **setMonitoringData[0].value** = *2* - **setMonitoringData[0].type** = *Periodic* ~~- **setMonitoringData[0].component.name** = *"EVSE"*~~ ~~- **setMonitoringData[0].component.evse.id** = *<Configured evseId>*~~ ~~- **setMonitoringData[0].variable.name** = *"AvailabilityState"*~~ - **setMonitoringData[0].component.name** = *<componentVariable.component.name>* - **setMonitoringData[0].component.evse.id** = *<componentVariable.component.evse.id>* - **setMonitoringData[0].variable.name** = *<componentVariable.variable.name>*
~~**3.** Charging Station generates **NotifyEventRequest** for EVSE #1::_AvailabilityState_ every *<Configured Clock Aligned MeterValues Interval>* seconds.~~ **3.** Charging Station sends a **NotifyEventRequest**	**4.** Test System responds with a **NotifyEventResponse**
Note(s): *Step 3 and 4 will repeat every 2 seconds*	

Tool validations
* Step 2: Message: **SetVariableMonitoringResponse** with: **setMonitoringResult[0].status** = *Accepted* ~~**setMonitoringResult[0].component.name** = *"EVSE"*~~ ~~**setMonitoringResult[0].component.evse.id** = *<Configured evseId>*~~ ~~**setMonitoringResult[0].variable.name** = *"AvailabilityState"*~~ **setMonitoringResult[0].component.name** = *<componentVariable.component.name>* **setMonitoringResult[0].component.evse.id** = *<componentVariable.component.evse.id>* **setMonitoringResult[0].variable.name** = *<componentVariable.variable.name>* **setMonitoringResult[0].attributeStatusInfo** is absent or **attributeStatusInfo.reasonCode** = *"NoError"*
* Step 3: ~~Message: a **NotifyEventRequest** message every *<Configured Clock Aligned MeterValues Interval>* seconds with:~~ Message: **NotifyEventRequest** every 2 seconds with: with an **eventData** element with: - **trigger** = *Periodic* ~~- **component.name** = *"EVSE"*~~ ~~- **component.evse.id** = *1*~~ ~~- **variable.name** = *"AvailabilityState"*~~ - **component.name** = *<componentVariable.component.name>* - **component.evse.id** = *<componentVariable.component.evse.id>* - **variable.name** = *<componentVariable.variable.name>*
Post scenario validations: N/A

6.2.24. Page 493 - (2024-09) - TC_N_41_CS - Set Variable Monitoring - Return to FactoryDefault

Moved preconfigured monitor to Prerequisite.

Test case name	**Set Variable Monitoring - Return to FactoryDefault**	
Test case Id	TC_N_41_CS	
...		
Prerequisite(s)	Charging Station supports Monitoring and a preconfigured monitor exists with *id* *<Preconfigured monitor id>* for *component* *EVSE* and *variable* *AvailabilityState* and *type* = *Delta* and *severity* = *<Preconfigured severity>*	
Before (Preparations)	**Configuration state:** N/a	
	Memory state: ~~a preconfigured monitor exists with *id <Preconfigured monitor id>* for *component EVSE* and *variable AvailabilityState* and *type = Delta* and *severity = <Preconfigured severity>*~~	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	...	

6.2.25. Page 482 - (2024-09) - TC_N_63_CS - Clear Customer Information - Clear and report - customerCertificate

Test case design top stop transaction was not correct for an ISO 15118 session.

Test case name	**Clear Customer Information - Clear and report - customerCertificate**
Test case Id	TC_N_63_CS
...	
Before (Preparations)	**Configuration State:** N/a
	Memory State: N/a
	Charging State: Execute **Reusable State** *EVConnectedPreSession* Execute **Reusable State** *Authorized15118* ~~Execute **Reusable State** *ParkingBayUnoccupied*~~

Test case name	**Clear Customer Information - Clear and report - customerCertificate**	
Main (Test scenario)	**Charging Station**	**CSMS**
	Note : The Charging Station receives a SessionStopReq(Terminate) message from the EV to finish the transaction.	
	2. The Charging Station responds with a **CustomerInformationResponse**	**1.** The OCTT sends a **CustomerInformationRequest** with - **report** *true* AND - **clear** *true* AND - **customerCertificate** *customer information used in the transaction*
	3. The Charging Station sends a **NotifyCustomerInformationRequest**	**4.** The OCTT responds with a **NotifyCustomerInformationResponse**
	Note(s): *- If **tbc** is True at Step 3 then step 3 and 4 will be repeated*	
	6. The Charging Station responds with a **CustomerInformationResponse**	**5.** The OCTT sends a **CustomerInformationRequest** with - **report** *true* AND - **clear** *false* AND - **customerCertificate** *customer information used in the transaction*
	7. The Charging Station sends a **NotifyCustomerInformationRequest**	**8.** The OCTT responds with a **NotifyCustomerInformationResponse**
	Note(s): *- If **tbc** is True at Step 7 then step 7 and 8 will be repeated*	
Tool validations	…	

6.2.26. Page 493 - (2024-09) - TC_N_41_CS - Set Variable Monitoring - Return to FactoryDefault

Moved preconfigured monitor to Prerequisite.

Test case name	**Set Variable Monitoring - Return to FactoryDefault**	
Test case Id	TC_N_41_CS	
…		
Prerequisite(s)	Charging Station supports Monitoring and a preconfigured monitor exists with *id* *<Preconfigured monitor id>* for *component* *EVSE* and *variable* *AvailabilityState* and *type* = *Delta* and *severity* = *<Preconfigured severity>*	
Before (Preparations)	**Configuration state:** N/a	
	Memory state: ~~a preconfigured monitor exists with *id <Preconfigured monitor id>* for *component EVSE* and *variable AvailabilityState* and *type = Delta* and *severity = <Preconfigured severity>*~~	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	…	

6.2.27. Page 495 - (2024-11) - TC_N_43_CS - Remove incorrect tool validation StatusInfo

Test case name	**Set Variable Monitoring - First SetMonitoringData and third SetMonitoringData are valid, but the second contains an out of range value**
Test case Id	TC_N_43_CS
…	…

Tool validations

* Step 2:
Message: **SetVariableMonitoringResponse** with (in arbitrary order):
setMonitoringResult[1] = {
- **status** = *Accepted*
- **type** = *UpperThreshold*
- **statusInfo** is absent or **statusInfo.reasonCode** = *"NoError"*
}
setMonitoringResult[2] = {
- **status** = *Rejected*
- **type** = *Delta*
- **statusInfo** is absent or **statusInfo.reasonCode** = "NoError" (Removed)
}
setMonitoringResult[3] = {
- **status** = *Accepted*
- **type** = *LowerThreshold*
- **statusInfo** is absent or **statusInfo.reasonCode** = *"NoError"*
}

Post scenario validations: - N/a

6.2.28. Page 555 - (2024-11) - Remove StatusNotificationRequest from Authorized reusable state Main B steps

Main B (Test scenario)	
Charging Station	**CSMS**
2. The Charging Station responds with a **RequestStartTransactionResponse**	**1.** The OCTT sends a **RequestStartTransactionRequest** with **idToken.idToken** *<Configured valid_idtoken_idtoken>* **idToken.type** *<Configured valid_idtoken_type>* **evseId** *<Configured evseId>*
3. The Charging Station sends an **AuthorizeRequest** Note(s): *- This step needs to be executed when **AuthCtrlr.AuthorizeRemoteStart** is true, unless (**AuthEnabled** is implemented with mutability ReadOnly AND the value is set to false) OR* *the **idToken** is cached.* *In case the **idToken** is used for a reservation, sending the **AuthorizeRequest** message is optional.*	**4.** The OCTT responds with an **AuthorizeResponse** with **idTokenInfo.status** *Accepted*
<Removed>	

Main B (Test scenario)	
5. The Charging Station sends a **TransactionEventRequest** Note(s): *- This step needs to be executed when **TxStartPoint** contains Authorized OR the transaction already started. So in the case **TxStartPoint** contains ParkingBayOccupancy or (EVConnected, in the case this testcase was initiated from state EVConnectedPreSession.)*	**6.** The OCTT responds with a **TransactionEventResponse** Note(s): *- The first TransactionEventRequest sent after authorization contains the idToken field. The TransactionEventResponse of this request message contains **idTokenInfo** with **status** Accepted*

6.2.29. Page 575 - (2024-09) - Reusable state RenewChargingStationCertificate expects a reconnection [784]

If a valid certificate is installed, then charging station must use it. This involves reconnecting to set up a new TLS with the new certificate. If the charging station does not do so automatically, then OCTT will force it by sending a Reset command.

State	**RenewChargingStationCertificate**	
System under test	Charging Station	
Description	The ChargingStationCertificate is renewed using A02/A03	
…		
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **TriggerMessageResponse**	**1.** The OCTT sends a **TriggerMessageRequest** With **requestedMessage** *SignChargingStationCertificate*
	3 The Charging Station sends a **SignCertificateRequest**	**4.** The OCTT responds with a **SignCertificateResponse** With **status** *Accepted*
	6. The Charging Station responds with a **CertificateSignedResponse**	**5.** The OCTT sends a **CertificateSignedRequest** With **certificateChain** *<Certificate generated from the received CSR from step 3 and signed by the provided CSMS Root certificate>* **certificateType** *ChargingStationCertificate*
	If the certificate is valid, then Charging Station should reconnect with the new certificate. OCTT waits some time for a reconnection, and if that does not occur, will send a Reset command to Charging Station to force a reconnection.	
	7. The Charging Station reconnects.	
	8 . *If the reconnect was forced by a Reset:* The Charging Station sends a **BootNotificationRequest**	**9.** OCTT responds with a **BootNotificationResponse** .

State	RenewChargingStationCertificate
Tool validations	* Step 2: Message: **TriggerMessageResponse** - **status** must be *Accepted* * Step 3: Message: **SignCertificateRequest** - **csr** must contain *<An CSR that meets the following requirements:* *When using RSA or DSA the key must be at least 2048 bits long.* *and when using elliptic curve cryptography the key must be at least 224 bits long.* *The received CSR must be transmitted as described in RFC 2986 and then encoded in Privacy-Enhanced Mail (PEM) format.>* * Step 6: Message: **CertificateSignedResponse** - **status** must be *Accepted* * Step 7: Charging Station must reconnect with new certificate.
	Post scenario validations: N/a

6.3. CSMS

6.3.1. Page 593 - (2024-09) - TC_A_11_CSMS - Reconnect using new client certificate

The testcase is missing steps to reconnect using the new client certificate.

Test case name	**Update Charging Station Certificate by request of CSMS - Success - Charging Station Certificate**
Test case Id	TC_A_11_CSMS
...	...

Main (Test scenario)	
Charging Station	**CSMS**
1. Execute **Reusable State** *RenewChargingStationCertificate*	
2. The OCTT disconnects its current connection and reconnects to the CSMS with the new certificate.	**3.** The CSMS accepts the incoming connection request using the new certificate.

6.3.2. Page 596 - (2024-09) - TC_A_14_CSMS - Update Charging Station Certificate by request of CSMS - Invalid certificate

SecurityEventNotification(InvalidChargingStationCertificate) has been added.

Test case name	**Update Charging Station Certificate by request of CSMS - Invalid certificate**	
Test case Id	TC_A_14_CSMS	
...		
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **TriggerMessageResponse** With **status** *Accepted*	**1.** The CSMS sends a **TriggerMessageRequest**
	3 The OCTT sends a **SignCertificateRequest** With **csr** *<Configured CSR>* **certificateType** *ChargingStationCertificate*	**4.** The CSMS responds with a **SignCertificateResponse**
	6. The OCTT responds with a **CertificateSignedResponse** With **status** *Rejected*	**5.** The CSMS sends a **CertificateSignedRequest**
	7. The OCTT sends a **SecurityEventNotificationRequest** with **type** = *InvalidChargingStationCertificate*	**8.** The CSMS responds with a **SecurityEventNotificationResponse**
Tool validations	...	

6.3.3. Page 597 - (2024-09) - TC_A_19_CSMS - Added additional information regarding the use of the client certificates

Added additional information regarding the use of the client certificates.

Test case name	**Upgrade Charging Station Security Profile - Accepted**
Test case Id	TC_A_19_CSMS
...	...

Before (Preparations)
Configuration State: N/a
Memory State: If configured <Security profile> is 2, then *RenewChargingStationCertificate* The OCTT uses this certificate during the TLS handshake when connecting with security profile 3.
Reusable State(s): N/a

6.3.4. Page 597 - (2024-09) - TC_A_19_CSMS - Removed validation of OcppCsmsUrl [O20-4355]

Validation of OcppCsmsUrl has been removed, because in some implementations the URL changes with the security profile.

Test case name	**Upgrade Charging Station Security Profile - Accepted**	
Test case Id	TC_A_19_CSMS	
…		
Main (Test scenario)	**Charging Station**	**CSMS**
	…	…
Tool validations	* Step 1: Message **SetNetworkProfileRequest** - **connectionData.messageTimeout** *<Configured messageTimeout>* - ~~**connectionData.ocppCsmsUrl** *<Configured ocppCsmsUrl>*~~ - **connectionData.ocppInterface** *<Configured ocppInterface>* - **connectionData.ocppTransport** *JSON* - **connectionData.ocppVersion** *OCPP20* - **connectionData.securityProfile** *<Configured securityProfile + 1>* * Step 3: Message **SetVariablesRequest** **setVariableData**: - **variable.name** = *"NetworkConfigurationPriority"* - **component.name** = *"OCPPCommCtrlr"* - **attributeValue** = *<contains configurationSlot provided at step 1>*	
	Post scenario validations: - N/a	

6.3.5. Page 637 - (2024-11) - TC_C_50_CSMS - Changed reference to configured valid idToken to a specific eMAID idToken

For ISO 15118 plug & charge the Charging Station always needs to use an eMAID idToken, which equals the CN of the configured contract certificate.

Test case name	**Authorization using Contract Certificates 15118 - Online - Local contract certificate validation - Accepted**
Test case Id	TC_C_50_CSMS
Use case Id(s)	C07
Requirement(s)	C07.FR.04
System under test	CSMS
Description	The Charging Station is able to authorize with contract certificates when it supports ISO 15118.
Purpose	To verify if the CSMS is able to validate the certificate hash data and the provided eMAID.

Test case name	**Authorization using Contract Certificates 15118 - Online - Local contract certificate validation - Accepted**
Prerequisite(s)	- The configured eMAID is known by the CSMS as valid. - The configured contract certificate is valid. - The CN of the configured contract certificate equals the configured eMAID. - iso15118CertificateHashData has a responder URL that points to an OCSP service for OCTT. - CSMS does not have a cached OCSP response for the contract certificate.

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is *EVConnectedPreSession*

Main (Test scenario)	
Charging Station	**CSMS**
1. The OCTT sends an **AuthorizeRequest** With **idToken.idToken** *<Configured eMAID>* **idToken.type** *eMAID* **iso15118CertificateHashData** contains *<hashes from configured (V2G) certificate chain*	…
…	

6.3.6. Page 639 - (2024-09) - TC_C_52_CSMS - TC does not use <Configured contract_certificate>

OCTT already has a keystore that contains the certificate. The pdf should not mention the <Configured contract_certificate> as the testcase does not use it

Test case name	**Authorization using Contract Certificates 15118 - Online - Central contract certificate validation - Accepted**	
Test case Id	TC_C_52_CSMS	
…		
Prerequisite(s)	- The configured eMAID is known by the CSMS as valid. - The ~~configured~~ contract certificate is signed by the configured V2GRoot or MORoot certificate at the CSMS. - Contract certificate has a responder URL that points to an OCSP service for OCTT. - CSMS does not have a cached OCSP response for the contract certificate.	
…		
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The OCTT sends an **AuthorizeRequest** With **idToken.idToken** *<Configured valid_idtoken_idtoken>* **idToken.type** *<Configured valid_idtoken_type>* **iso15118CertificateHashData** is absent **certificate** *from keystore*	**2.** The CSMS sends an OCSP request to responder URL of **certificate** to check validity
…		

6.3.7. Page 712 - (2024-09) - TC_I_01_CSMS - Show EV Driver running total cost

Test case name	**Show EV Driver running total cost during charging - costUpdatedRequest**
Test case Id	TC_I_01_CSMS
…	

Test case name	**Show EV Driver running total cost during charging - costUpdatedRequest**	
Main (Test scenario)	**Charging Station**	**CSMS**
	…	
	7. The OCTT sends a **TransactionEventRequest** With **triggerReason** is *MeterValuePeriodic* **eventType** is *Updated* **timestamp** *<The intervals between the timestamps of the received Meter Value messages equals the configured sampled Meter Values interval>.* **sampledValue.context** is *Sample.Periodic* *Note(s):* _- This step will be executed every _<Configured sampled Meter Values interval> - The OCTT will end the testcase after two MeterValues.	**8.** The CSMS responds with a **TransactionEventResponse**
	…	
Tool validations	…	
	Post scenario validations: - N/a	

6.3.8. Page 728 - (2024-09) - TC_K_03_CSMS - Not requiring validFrom/To fields in charging profile [O20-4592] and chargingProfileKind must be Absolute [O20-4591]

Test case name	**Set Charging Profile - ChargingStationMaxProfile**	
Test case Id	TC_K_03_CSMS	
…		
Main (Test scenario)	**Charging Station**	**CSMS**
	…	…
Tool validations	* Step 1: Message **SetChargingProfileRequest** **evseId** *0* AND **chargingProfile.stackLevel** *<Configured stackLevel>* AND **chargingProfile.chargingProfilePurpose** *ChargingStationMaxProfile_* AND **chargingProfile.chargingProfileKind** *Absolute* ~~OR *Relative*~~ **chargingProfile.chargingSchedule.chargingRateUnit** *<Configured ChargingRateUnit>* **chargingProfile.chargingSchedule.duration** *<Configured duration>* **chargingProfile.chargingSchedule.chargingSchedulePeriod.startPeriod** *0* **chargingProfile.chargingSchedule.chargingSchedulePeriod.limit** *8.0* or *8000.0* **chargingProfile.chargingSchedule.chargingSchedulePeriod.numberPhases** *<Configured numberPhases>* where *<Configured numberPhases>* not *3* OR **chargingProfile.chargingSchedule.chargingSchedulePeriod.numberPhases** *<Configured numberPhases>* or *<omit>* where *<Configured numberPhases> 3* ~~**chargingProfile.validFrom** *<Not omitted>*~~ ~~**chargingProfile.validTo** *<Not omitted>*~~ **chargingProfile.chargingSchedule.startSchedule** *<Not omitted>*	
	Post scenario validations: - N/a	

6.3.9. Page 733 - (2024-09) - TC_K_10_CSMS - Not requiring validFrom/To fields in charging profile [O20-4592]

Test case name	**Set Charging Profile - TxDefaultProfile - All EVSE**	
Test case Id	TC_K_10_CSMS	
…		
Main (Test scenario)	**Charging Station**	**CSMS**
	…	…
Tool validations	* Step 1: Message **SetChargingProfileRequest** **evseId** *0* AND **chargingProfile.stackLevel** *<Configured stackLevel>* AND **chargingProfile.chargingProfilePurpose** *TxDefaultProfile* AND **chargingProfile.chargingProfileKind** *Absolute* AND ~~**chargingProfile.validFrom** *<Not omitted>* AND~~ ~~**chargingProfile.validTo** *<Not omitted>* AND~~ **chargingProfile.chargingSchedule.startSchedule** *<Not omitted>* AND **chargingProfile.chargingSchedule.chargingRateUnit** *<Configured ChargingRateUnit>* AND **chargingProfile.chargingSchedule.chargingSchedulePeriod.startPeriod** *0* AND **chargingProfile.chargingSchedule.duration** *<Configured duration>* **chargingProfile.chargingSchedule.chargingSchedulePeriod.limit** *6.0* or *6000.0* AND **chargingProfile.chargingSchedule.chargingSchedulePeriod.numberPhases** *<Configured numberPhases>* where *<Configured numberPhases>* not *3* OR **chargingProfile.chargingSchedule.chargingSchedulePeriod.numberPhases** *<Configured numberPhases>* or *<omit>* where *<Configured numberPhases> 3*	
	Post scenario validations: - N/a	

6.3.10. Page 734 - (2024-09) - TC_K_15_CSMS - Not requiring validFrom/To fields in charging profile [O20-4592]

Test case name	**Set Charging Profile - Not Supported**
Test case Id	TC_K_15_CSMS
…	

Test case name	**Set Charging Profile - Not Supported**	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with RPC Framework: CALLERROR: NotSupported.	**1.** The CSMS sends a **SetChargingProfileRequest** with: **evseId** *<Configured evseId>* AND **chargingProfile.stackLevel** *<Configured stackLevel>* AND **chargingProfile.chargingProfilePurpose** *TxDefaultProfile* AND **chargingProfile.chargingProfileKind** *Absolute* AND ~~**chargingProfile.validFrom** *<Not omitted>* AND~~ ~~**chargingProfile.validTo** *<Not omitted>* AND~~ **chargingProfile.chargingSchedule.startSchedule** *<Not omitted>* AND **chargingProfile.chargingSchedule.chargingRateUnit** *<Configured ChargingRateUnit>* AND **chargingProfile.chargingSchedule.chargingSchedulePeriod.startPeriod** *0* AND **chargingProfile.chargingSchedule.duration** *<Configured duration>* **chargingProfile.chargingSchedule.chargingSchedulePeriod.limit** *6.0* or *6000.0* AND **chargingProfile.chargingSchedule.chargingSchedulePeriod.numberPhases** *<Configured numberPhases>*
Tool validations	...	

6.3.11. Page 753 - (2024-09) - TC_K_55_CSMS, TC_K_57_CSMS, TC_K_58_CSMS, TC_K_59_CSMS Removed triggerReason = ChargingRateChanged [776]

A trigger reason ChargingStateChange must only be sent, when an external actor (not CSMS) changes the charging rate. Therefore, removed the sending of a triggerReason=ChargingStateChanged by OCTT. This does not affect tool validations, but it was incorrect behavior of OCTT.

The step that sends a TransactionEventRequest with triggerReason=ChargingRateChanged has been removed from:

- TC_K_55_CSMS
- TC_K_57_CSMS
- TC_K_58_CSMS
- TC_K_59_CSMS

6.3.12. Page 760 - (2024-11) - TC_K_70_CSMS - Updated tool validation chargingProfiles and added preparation step.

We require a CSMS to install multiple ChargingProfiles with the same purpose and for the same connectorId with a different stackLevel. However there are CSMSs that do the stacking themselves and are unable to do this. Therefore it was decided to use different purposes instead.

Test case name	**Set Charging Profile - Multiple Profiles**
Test case Id	TC_K_70_CSMS
...	
Before (Preparations)	...
	Charging State: State is *EnergyTransferStarted*

Test case name	**Set Charging Profile - Multiple Profiles**	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **SetChargingProfileResponse** with **status** *Accepted*	**1.** The CSMS sends a **SetChargingProfileRequest** with **chargingProfilePurpose** *TxDefaultProfile*
	4. The OCTT responds with a **SetChargingProfileResponse** with **status** *Accepted*	**3.** The CSMS sends a **SetChargingProfileRequest** with **chargingProfilePurpose** *ChargingStationMaxProfile*
Tool validations	* Step 1: Message **SetChargingProfileRequest** **chargingProfile.chargingProfilePurpose** *TxDefaultProfile* * Step 3: Message **SetChargingProfileRequest** **chargingProfile.id** *<different id for both chargingProfiles>* **chargingProfile.chargingProfilePurpose** *ChargingStationMaxProfile*	
	Post scenario validations: - N/a	

6.3.13. Page 830 - (2024-11) - TC_N_46_CSMS - Updated tool validation sendLocalListRequest

Test case name	**Clear Customer Information - Update Local Authorization List**	
Test case Id	TC_N_46_CSMS	
…		
Before (Preparations)	…	
	Memory State: A local authorization list with *<Configured valid_idtoken_idtoken>* is configured.	
	…	
Main (Test scenario)	**Charging Station**	**CSMS**
	…	…
Tool validations	* Step 1: Message **CustomerInformationRequest** - **report** *true* AND - **clear** *true* AND - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>* * Step 5: Message **SendLocalListRequest** - **updateType** *Differential* - **versionNumber** *<Bigger than currently configured in OCTT>* - **localAuthorizationList** *<Contains only the configured valid_idtoken_idtoken, without idTokenInfo>* OR - **updateType** *Full* - **localAuthorizationList** *<Does NOT contain configured valid_idtoken_idtoken>*	
	Post scenario validations: - N/a	