OCPP 2.0.1 Edition 2

Errata 2023-12

Table of Contents

Disclaimer

Version History

Version	Date	Description
2023-12	2023-12-18	Includes new errata for Part 1, Part 2, Part 4, Part 5 and Part 6 of OCPP 2.0.1
1.0 → 2023-06	2023-06-30	Release of Edition 2 Errata v1.0 (v1) changes are now marked as (2023-06)

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 2.

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

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

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 2023-12 of the errata. The errata of this version are marked with "(2023-12)" in the section title. Please note that due to a new versioning scheme for the errata document, the errata that were previously marked with "(v1)" are now marked with "(2023-06)"

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 part 0.

1. Part 1

1.1. Device Model: Addressing Components and Variables

1.1.1. Page 6 - (2023-12) - section 4.1 Components: Clarification about tiers for EVSE/Connector components

It was not made explicit in the text, that the EVSE **component** must be addressed as being part of the EVSE **tier**. Similarly, a Connector **component** must be at the connector **tier**. This is shown correctly in the tables for "Basic home charging example configuration" and "Public home charger example configuration", but was not mentioned explicitly.

Therefore, this sentence is extended, as follows:

After this text	*ChargingStation* (TopLevel), *EVSE*, and *Connector* represent the three major "tiers" of a Charging Station, and constitute an implicit "location-based" addressing scheme that is widely used in many OCPP data structures.
Add new text	Each "tier" has a component of the same name, which represents the tier. For example, EVSE 1 on a Charging Station is represented by the component named "EVSE" (no instance name) with "*evseId* = 1". In the same manner, Connector 1 on EVSE 1 is represented by the component named "Connector" (no instance name) with "*evseId* = 1, *connectorId* = 1".

1.1.2. Page 10 - (2023-12) - GetBaseReport supported 'ReportBases' [350]

The table with use cases that are part of a Minimum Device Model implementation has an error for *"B07 Get Base Report"*. Replace the text as follows:

Old text	GetBaseReport message MUST be implemented and MUST support all 3 ‘ReportBases’.
New text	GetBaseReport message MUST be implemented and MUST support ConfigurationInventory and FullInventory.

2. Part 2

2.1. General

2.2. Use case A Security

2.2.1. Page 23 - (2023-06) Requirement A00.FR.316: Make clear that InvalidTLSVersion must be queued [689]

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

Changed requirement

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

Page 25 - Requirement A00.FR.419

Changed requirement

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

2.3. Use case B Provisioning

2.3.1. Page 51 - (2023-12) Requirement B03.FR.08 incorrect, and B03.FR.03 rephrased [712]

Requirement B03.FR.08 suggests that CSMS can send a TriggerMessage(BootNotification) after it has rejected the Charging Station. This is not possible.

	ID	Precondition	Requirement definition
Old	B03.FR.03	While in the status *Rejected*.	The CSMS SHALL NOT initiate any messages.
New	B03.FR.03	When the CSMS has Rejected the BootNotificationRequest from the Charging Station.	The CSMS SHALL NOT initiate any messages.
Old	B03.FR.08	B03.FR.03 AND CSMS sends a message that is not a TriggerMessageRequest(requestedMessage = *BootNotification*)	Charging Station SHALL respond with RPC Framework: CALLERROR: SecurityError.
New	B03.FR.08	B03.FR.03 AND CSMS sends a message that is not a response to a BootNotificationRequest from Charging Station	Charging Station SHALL respond with RPC Framework: CALLERROR: SecurityError.

2.3.2. Page 51 - (2023-12) Typo in B03.FR.06 [736]

	ID	Precondition	Requirement definition
Old	B03.FR.06	If the interval in the BootNotificationResponse is greater than 0, and the status is other than *Accepted*	The Charging Station SHALL send a BootNotificationRequest after the set interval has past.
New	B03.FR.06	If the interval in the BootNotificationResponse is greater than 0, and the status is other than *Accepted*	The Charging Station SHALL send a BootNotificationRequest after the set interval has passed.

2.3.3. Page 61 - (2023-06) Requirement B08.FR.19 and N02.FR.15 are ambiguous w.r.t. evse and instance wildcards [676]

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

Delete requirement

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

The following new requirements replace B08.FR.19:

New requirements

ID	Precondition	Requirement definition
B08.FR.22	B08.FR.11 AND When Charging Station receives a GetReportRequest with a *component* in a *componentVariable* element that has a *component.evse.id*, but *component.evse.connector* is missing	The Charging Station SHALL report the component(s) with this *component.name*, *component.instance* and *component.evse.id* for every *component.evse.connector*, whilst taking into account B08.FR.24.

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

Page 319 - N02.FR.15

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

Delete requirement

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

New requirements

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

2.3.4. Page 62 - (2023-06) Use case B09/B10: Extended scenario description [683]

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

Use case B09

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

No.	Type	Description
1	Name	Setting a new NetworkConnectionProfile.
2	ID	B09
	Functional block	B. Provisioning
3	Objectives	To enable the CSMS to update the connection details on the Charging Station.
4	Description	The CSMS updates the connection details on the Charging Station. For instance in preparation of a migration to a new CSMS. After completion of this use case, the Charging Station to CSMS connection data has been updated.
	Actors	Charging Station, CSMS
	Scenario description	**A Charging Station supports at least two network configuration slots, that are identified by a number. The available slot numbers are reported by the Charging Station in the *valuesList* of variable NetworkConfigurationPriority.** **For example: *valuesList* = "0,1,2" in the base report tells CSMS that three configuration slots, numbered 0, 1 and 2, are available (but not necessarily set).** **The configuration slot number that is used for the active connection is reported by variable OCPPCommCtrlr.ActiveNetworkProfile.** 1. The CSMS sends a SetNetworkProfileRequest PDU containing an updated connection profile **and a *configurationSlot* out of the *valuesList* of NetworkConfigurationPriority.** 2. The Charging Station receives the PDU, validates the content and stores the new data 3. The Charging Station responds by sending a SetNetworkProfileResponse PDU, with status *Accepted*
5	Prerequisites	The data supplied by the CSMS matches the Charging Station's capabilities
6	Postcondition(s)	The Charging Station was able to store the new connection data

Requirement for configuration slots

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

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

New requirements

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

Use case B10

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

No.	Type	Description
1	Name	Migrate to new CSMS, using a different NetworkConnectionProfile.
2	ID	B10
	Functional block	B. Provisioning
3	Objectives	After completion of this use case, the Charging Station connects to a new CSMS.
4	Description	This use case describes how a Charging Station can be instructed to connect to a new CSMS, by changing the order of NetworkConnectionProfiles in `NetworkConfigurationPriority`.

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

2.4. Use case C Authorization

2.4.1. Page 90 - (2023-06) C07 requirements for *certificateStatus* missing [680]

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

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

New requirements

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

ID	Precondition	Requirement definition	Note
C07.FR.17	C07.FR.04 AND the certificate chain (provided in *certificate* or *iso15118CertificateHashData*) cannot be verified or is invalid	CSMS SHALL return an AuthorizationResponse containing a *certificateStatus* = `CertChainError` and an *idTokenInfo.status* = `Invalid`	If certificate is cannot be verified, then status of *idToken* is reported as invalid.

Page 408 - AuthorizeCertificateStatusEnumType

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

Updated text in **bold**:

AuthorizeCertificateStatusEnumType

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

2.4.2. Page 97 - (2023-06) Requirement C10.FR.06 needs to be removed [685]

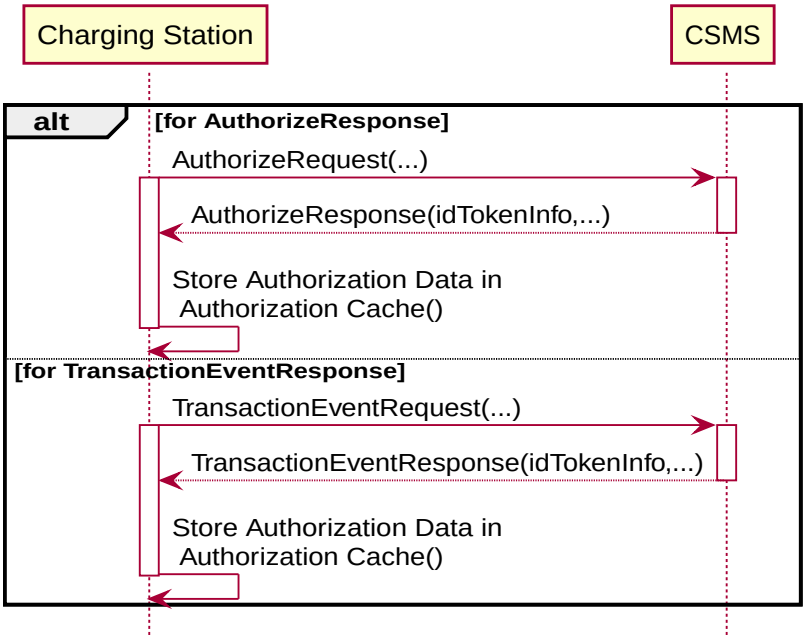
Requirement C10.FR.06 is an invalid requirement, because the ReserveNowRequest does not contain IdTokenInfo, so there is no information to update the Authorization Cache with.

Deleted requirement

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

Page 96 - Update sequence diagram

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



2.4.3. Page 102 - (2023-06) Requirement C13.FR.04 enhanced [701]

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

Changed requirement

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

2.5. Use Case E Transactions

2.5.1. Page 147 - (2023-06) Use case E07 - Scenario description step order incorrect [704]

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

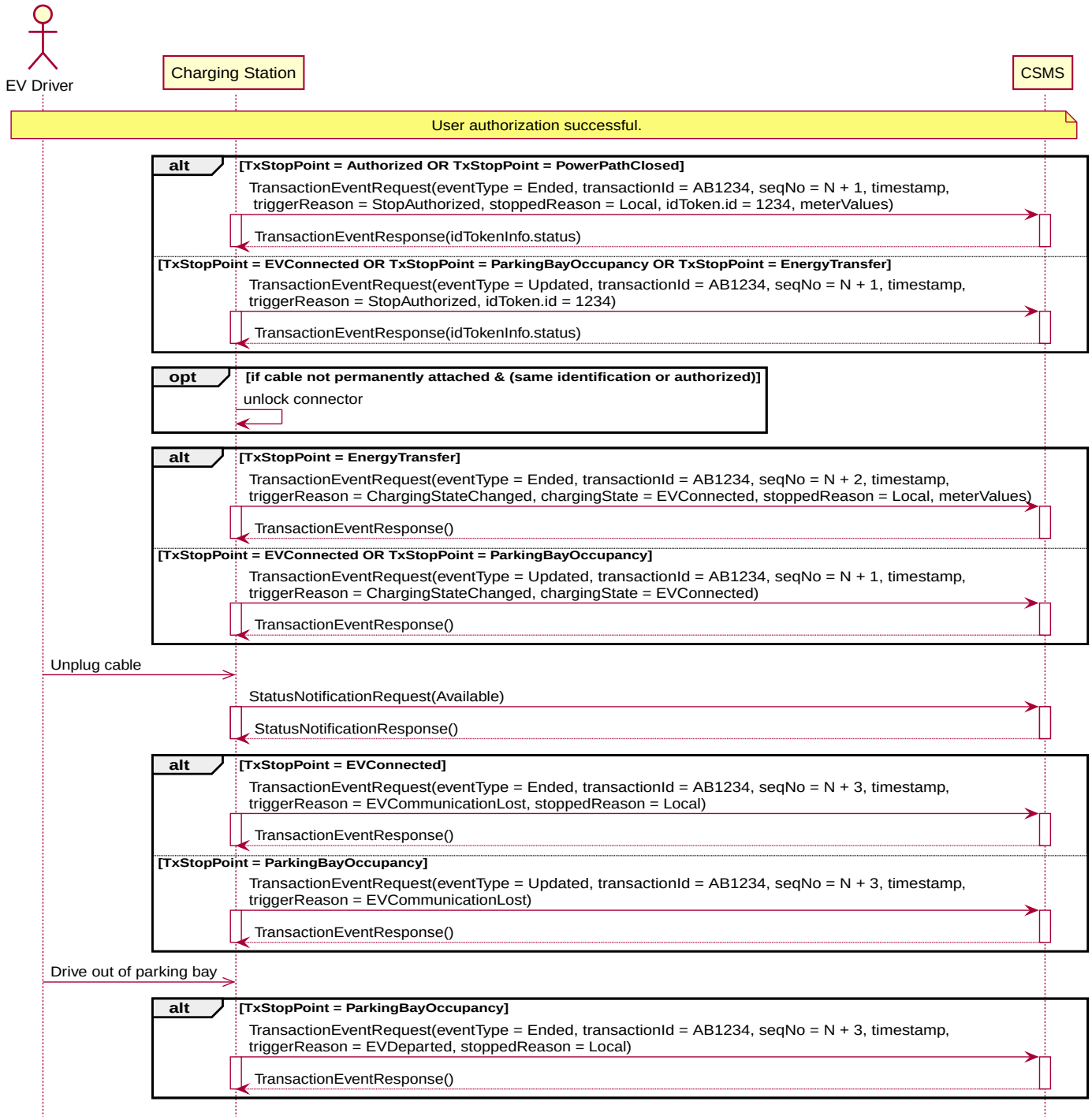
	No.	Type	Description
Old text		*Scenario description* TxStopPoint = `Authorized` (or `PowerPathClosed`)	**1.** The EV Driver presents IdToken a second time to end charging. **2.** The Charging Station sends a TransactionEventRequest (eventType = Ended) with *triggerReason* = `StopAuthorized` and *stoppedReason* = `Local`. **3.** The CSMS responds with a TransactionEventResponse. **4.** The Charging Station stops the energy transfer and if the cable is not permanently attached, the Charging Station unlocks the cable.

	No.	Type	Description
New text		*Scenario description* 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.

2.5.2. Page 148 - (2023-06) Use case E07: Wrong triggerReason shown in sequence diagram fig. 56 [687]

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

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



2.5.3. Page 150 - (2023-06) Use case E07: Clarify 'normal' and 'correct' for *stoppedReason* [693]

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

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

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

Page 403 - TransactionType field *stoppedReason*

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

TransactionType

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

2.6. Use Case F Remote Control

2.6.1. Page 180 - (2023-06) Requirement F03.FR.03 contains wrong precondition [700]

The precondition of requirement F03.FR.03 was incorrectly merged from Errata v2 into Edition 2, and the associated Note was not relevant for this situation.
It needs to be changed as follows:

Changed requirement

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

2.6.2. Page 187 - (2023-06) Requirement F06.FR.12 is too strict [707]

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

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

Changed requirement

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

2.7. Use Case G Availability

2.7.1. Page 192 - (2023-06) G01.FR.08 contradicts H01.FR.24 [692]

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

Table 1. G01 - Requirements

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

2.8. Use Case H Reservation

2.8.1. Page 205 - (2023-06) Missing option to send NotifyEvent instead of StatusNotification [699]

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

Changed requirements

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

Page 203 - (2023-06) Added option to use case description to send NotifyEventRequests

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

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

	Scenario description	1. EV Driver asks the CSMS to reserve a specific EVSE at the Charging Station. 2. The CSMS sends ReserveNowRequest with a EVSE to a Charging Station. 3. Upon receipt of ReserveNowRequest, the Charging Station responds with ReserveNowResponse with status *Accepted*. 4. The Charging Station sends StatusNotificationRequest with the status *Reserved* for all Connectors of that EVSE. 5. The CSMS responds with StatusNotificationResponse to the Charging Station.
	Alternative scenario description	**Steps 1, 2 and 3 as above.** **4. Instead of a StatusNotificationRequest a Charging Station can send a NotifyEventRequest with *trigger* = `Delta` for *component.name* = "Connector" and the EVSE number in *evse.id* and the connector number in *evse.connectorId*, *variable* = "AvailabilityState" and *actualValue* = "Reserved".** **5a. Optionally, Charging Station can also report a NotifyEventRequest for *component* = "EVSE", *variable* = "AvailabilityState" and *actualValue* = "Reserved" , and when applicable, also report this for *component* = "ChargingStation".**
	Prerequisite(s)	The specified EVSE of the Charging Station has status *Available*
	Postcondition(s)	Successful postcondition: The Charging Station has accepted the ReserveNowRequest AND sent StatusNotificationRequests with status *Reserved*. Failure postcondition: The Charging Station has rejected the ReserveNowRequest OR The Charging Station has NOT sent StatusNotificationRequests with status *Reserved*.

2.8.2. Page 209 - (2023-06) Remark about authorization in use case H03 [711]

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

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

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

2.8.3. Page 210 - (2023-06) Requirement H03.FR.08 is not clear about groupIdToken lookup [684]

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

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

Changed requirement

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

2.8.4. Page 210 - (2023-12) Transaction can start even when connector is Reserved [735]

It is not sufficiently clear from use case H03 that a transaction on a reserved connector will be started at the time of cable plug-in or occupancy of parking bay when TxStartPoint is `EVConnected` or `ParkingBayOccupancy`. However, only the IdToken (or groupIdToken) that matches the reservation can be authorized. Non-reserved IdTokens will therefore not be able to charge.

This is clarified by adding the following requirements:

Page 210 - H03

New requirement

ID	Precondition	Requirement definition
H03.FR.09	When an *idToken* or *groupIdToken* is presented that matches a reservation	Charging Station SHALL consider the reservation to be used (consumed)
H03.FR.10	H03.FR.09 AND Connector associated with reservation has status `Reserved`	Charging Station SHALL set connector status to `Available` if no cable has been plugged-in, or `Occupied` if a cable has already been plugged-in.

Page 196 - G03

New requirement

ID	Precondition	Requirement definition	Note
G03.FR.09	The connector is Reserved when an EV is connecting AND EV driver has not presented an IdToken matching the reservation	Connector status SHALL not change.	Connector stays reserved until IdToken matching reservation is presented or reservation expires.

2.9. Use Case J Meter Values

2.9.1. Page 228 - (2023-06) Requirement J01.FR.14 is unclear that meter values for all EVSEs must be sent [674]

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

Changed requirement

	ID	Precondition	Requirement definition	Note
Old text	J01.FR.14	When configured to send MeterValuesRequest, See: Meter Values - Configuration	The Charging Station SHALL send MeterValuesRequest messages to the CSMS as configured.	

	ID	Precondition	Requirement definition	Note
New text	J01.FR.14	When configured to send MeterValuesRequest, See: Meter Values - Configuration	The Charging Station SHALL send MeterValuesRequest messages to the CSMS as configured **in Meter Values - Configuration, for all *evseIds*, locations and phases for which a configured measurand is supported.**	**It is allowed to report the measurands for EVSEs with an ongoing transaction using the TransactionEventRequest message. It is possible that certain measurands are not available for every location. For example, *evseId* = 0 (grid meter) will not have a "Current.Offered" or "SoC" measurand.**

2.9.2. Page 230 - (2023-06) Requirement J02.FR.10 refers to all TransactionEventRequest messages, but should be specific to only eventType = Updated [705]

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

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

Changed requirement

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

2.9.3. Page 231 - (2023-06) J01 misses requirement that meter value must be for current transaction [673]

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

New requirement

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

2.10. Use Case K Smart Charging

2.10.1. Page 238 - (2023-06) Text in section 3.3 does not match ChargingProfileKindEnumType description [708]

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

Below is the updated text shown in bold:

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

2.10.2. Page 276 - (2023-12) Requirement K15.FR.15 has wrong precondition [716]

Requirement K15.FR.15 should refer to the moment when EV sends charging needs, which is K15.FR.01.

	ID	**Precondition**	**Requirements**	**Note**
Old	K15.FR.15	K15.FR.03 AND Charging Station is offline	The Charging Station SHALL use the TxDefaultProfile (if present) and generate a charging schedule within the limits of its composite schedule.	
New	K15.FR.15	K15.FR.01 AND Charging Station is offline	The Charging Station SHALL use the TxDefaultProfile (if present) and generate a charging schedule within the limits of its composite schedule.	

2.11. Use Case L FirmwareManagement

2.11.1. Page 287 - (2023-06) Improved title of figure 119 [695]

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

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

2.12. Use Case M ISO 15118 CertificateManagement

2.12.1. Page 310 - (2023-06) M04.FR.07 has an incorrect requirement definition [703]

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

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

2.13. Use Case N Diagnostics

2.13.1. Page 317 - (2023-06) N01.FR.10 not clear when to report UploadFailure [696]

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

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

2.13.2. Page 331 - (2023-06) Requirement N09.FR.04 has been rephrased [688]

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

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

New requirement

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

2.14. Messages

2.14.1. Page 353 - (2023-06) Clarification for use of *certificate* and *iso15118CertificateHashData* in AuthorizeRequest [675]

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

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

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

AuthorizeRequest

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

2.14.2. Page 381 - (2023-06) Updated description for idToken in TransactionEventRequest [709]

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

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

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

2.15. Data Types

2.15.1. Page 386 - (2023-06) issuerKeyHash in CertificateHashDataType must be type identifierString [691]

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

Changed field type for issuerKeyHash:

CertificateHashDataType

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

2.15.2. Page 396 - (2023-06) NetworkConnectionProfileType [683]

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

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

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

Changed descriptions in NetworkConnectionProfileType

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

2.15.3. Page 396 - (2023-12) NetworkConnectionProfileType [713]

The description of *ocppCsmsUrl* does not make clear that it is the URL **without** the charging station identity.

Changed description in NetworkConnectionProfileType

Field Name	Field Type	Card.	Description
ocppCsmsUrl	string[0..512]	1..1	Required. URL of the CSMS that this Charging Station communicates with, without the Charging Station identity part. The SecurityCtrlr.Identity field is appended to *ocppCsmsUrl* to provide the full websocket URL.

2.16. Enumerations

2.16.1. Page 419 - (2023-06) Description for idTokenEnumType MacAddress [664]

A description is missing for value `MacAddress` of IdTokenEnumType.

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

2.17. Referenced Components and Variables

2.17.1. Page 436 - (2023-12) Incorrectly referencing unit = "seconds" instead of "s" [726]

There are a number of variables that have "unit = seconds", because it refers to an interval or timeout in seconds. The official unit for seconds, however, is "s" as is stated in Appendix 2 "Standardized Units of Measure". Since this may be confusing, the field **unit** must be changed in to "s" for all these variables.

This affects the following list of variables:

- DefaultMessageTimeout
- HeartbeatInterval
- OfflineThreshold
- MessageAttemptIntervalTransactionEvent
- WebSocketPingInterval
- TimeAdjustmentReportingThreshold
- CertSigningWaitMinimum
- AuthCacheLifeTime
- EVConnectionTimeout
- SampledDataTxEndedInterval
- SampledDataTxUpdatedInterval
- AlignedDataInterval
- AlignedDataTxEndedInterval

NOTE The field "unit" is only for information to CSMS. The description of the variables already makes clear that it is about seconds.

2.17.2. Page 436 - (2023-06) Websocket-related variables in Part 4 [690]

Add the following note below section heading "General":

NOTE WebSocket-related variables are described in "OCPP-2.0.1 Part 4 JSON over WebSockets".

Page 439 - 2.1.13 WebSocketPingInterval

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

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

This configuration variable is described in "OCPP-2.0.1 Part 4 JSON over WebSockets".

2.17.3. Page 444 - (2023-06) SecurityCtrlr.BasicAuthPassword and Identity should have dataType=string

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

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

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

`BasicAuthPassword`

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

Required	no		
Component	**componentName**	SecurityCtrlr	
Variable	**variableName**	BasicAuthPassword	
	variableAttributes	**mutability**	WriteOnly
	variableCharacteristics	**dataType**	***string***
		maxLimit	40 (Max length of the BasicAuthPassword)
Description	The basic authentication password is used for HTTP Basic Authentication. The password SHALL be a randomly chosen passwordString with a sufficiently high entropy, consisting of minimum 16 and maximum 40 characters (alphanumeric characters and the special characters allowed by passwordString). The password SHALL be sent as a UTF-8 encoded string (NOT encoded into octet string or base64). This configuration variable is write-only, so that it cannot be accidentally stored in plaintext by the CSMS when it reads out all configuration variables. This configuration variable is required unless only "security profile 3 - TLS with client side certificates" is implemented.		

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

`Identity`

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

2.17.4. Page 452 - (2023-06) Incomplete description TxStopPoint Authorized and PowerPathClosed [704]

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

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

2.6.6.2 TxStopPoint values

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

2.18. Appendix 1

2.18.1. Page 2 - (2023-06) InvalidFirmwareSignature/SigningCertificate are critical security events [682]

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

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

2.19. Appendix 3

2.19.1. Page 9 - (2023-06) OCPPCommCtrlr.ActiveNetworkProfile must be of type integer [697]

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

OCPPCommCtrlr

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

2.19.2. Page 10 - (2023-06) SecurityCtrlr.BasicAuthPassword and Identity should have dataType=string [698]

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

SecurityCtrlr

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

2.20. Appendix 5

2.20.1. Page 36 - (2023-12) ReasonCodes *MissingDeviceModelInfo* and *InvalidMessageSequence* exceed 20 chars [720]

ReasonCodes *MissingDeviceModelInfo* and *InvalidMessageSequence* exceed 20 characters of *reasonCode* field. These needed to be shortened.

Old reason code name	**New reason code name**
MissingDeviceModelInfo	MissingDevModelInfo
InvalidMessageSequence	InvalidMessageSeq

Page 276 - Requirement K15.FR.17

ReasonCode `InvalidMessageSequence` is referenced in K15.FR.17 and needs to be updated.

Changed requirement

	ID	**Precondition**	**Requirements**	**Note**
Old	K15.FR.17	When Charging Station receives a SetChargingProfileRequest immediately after the transaction has started and before it has sent the NotifyEVChargingNeedsRequest to CSMS	The Charging Station SHOULD respond with SetChargingProfileResponse with *status* = `Rejected` and a *statusInfo* with *reasonCode*= `InvalidMessageSequence`.	CSMS sent profile too early. It does not harm if CS accepts the charging profile instead of rejecting it, as long as it sends a charging profile again when it receives the NotifyEVChargingNeedsRequest.
New	K15.FR.17	When Charging Station receives a SetChargingProfileRequest immediately after the transaction has started and before it has sent the NotifyEVChargingNeedsRequest to CSMS	The Charging Station SHOULD respond with SetChargingProfileResponse with *status* = `Rejected` and a *statusInfo* with *reasonCode*= `InvalidMessageSeq`.	CSMS sent profile too early. It does not harm if CS accepts the charging profile instead of rejecting it, as long as it sends a charging profile again when it receives the NotifyEVChargingNeedsRequest.

3. Part 3

Currently no new errata for OCPP 2.0.1 part 3.

4. Part 4

4.1. Page 8 - (2023-12) - section 3.1.2. No OCPP version in endpoint URL [732]

If websocket protocol negotiotion is to be used, the OCPP version should not be part of the endpoint URL. Therefore, the following paragraph in section 3.1.2 needs to be changed.

Changed text

3.1.2 OCPP version

The OCPP version(s) MUST be specified in the Sec-Websocket-Protocol field. This SHOULD be one or more of the following values:

Table 2. OCPP Versions

OCPP version	WebSocket subprotocol name
1.2	ocpp1.2
1.5	ocpp1.5
1.6	ocpp1.6
2.0	ocpp2.0
2.0.1	ocpp2.0.1

The ones for OCPP 1.2, 1.5, 1.6, 2.0 and 2.0.1 are official WebSocket subprotocol name values. They are registered as such with IANA.

Note that OCPP 1.2 and 1.5 are in the list. Since the JSON over WebSocket solution is independent of the actual message content the solution can be used for older OCPP versions as well. Please keep in mind that in these cases the implementation should preferably also maintain support for the SOAP based solution to be interoperable.

~~It is considered good practice to include the OCPP version as part of the OCPP-J endpoint URL string. If you run a web service that can handle multiple protocol versions on the same OCPP-J endpoint URL this is not necessary of course.~~ 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 .

4.2. Page 10 - (2023-12) - Section 4.1.4. The message ID must be unique [702]

The text in section 4.1.4 uses the wording *"on the same WebSocket connection"*. This can, however, be interpreted in multiple ways. It was intended to mean that that the messageId must be different from all messageIds previously used by the same sender for any other CALL message on any WebSocket connection using the same unique Charging Station identifier. The current wording seems to indicate that it may use the same messageId after every reconnect, however this may cause major issues. Especially when looking at the OCPP message queuing mechanisms.

Changed text:

4.1.4 The message ID

The message ID serves to identify a request. A message ID for any CALL message MUST be different from all message IDs previously used by the same sender for any other CALL message on any WebSocket connection using the same unique Charging Station identifier. A message ID for a CALLRESULT or CALLERROR message MUST be equal to that of the CALL message that the CALLRESULT or CALLERROR message is a response to.

Table 3. Unique Message ID

Name	Datatype	Restrictions
messageId	string[36]	Unique message ID, maximum length of 36 characters, to allow for UUIDs/GUIDs

5. Part 5

5.1. List of test cases

5.1.1. Page 11 - (2023-12) - TC_B_08_CS should not be tested

This test case tests requirement B06.FR.05, which is not a Charging Station requirement. The limit must be respected (not tested) by the CSMS / OCTT.

	OCTT Id	OCPP Compliance Testing Tool scenario	Conf. Test for Charging Station	Conf. test for CSMS	Condition / remark	Feature no.	Feature
Removed	TC_B_08	limit to maximum number of values	C		If the Charging Station supports BytesPerMessageGetVariables	ORS-5	BytesPerMessageGetVariables

5.1.2. Page 13 - (2023-12) - TC_B_50_CS - Testcase not only applicable for AdditionalRootCertificateCheck = true

This test case was conditional for feature AdditionalRootCertificateCheck, however this can always be performed (no relation with AdditionalRootCertificateCheck) Due to the importance of the functionality, the condition has been removed and the testcase has become mandatory.

	OCTT Id	OCPP Compliance Testing Tool scenario	Conf. Test for Charging Station	Conf. test for CSMS	Condition / remark	Feature no.	Feature
Old text	TC_B_50	Success - New CSMS Root - New CSMS	C		For CS: at least two configuration slots for networkConnectionProfiles must be supported	AS-2	Additional Root Certificate check
New text	TC_B_50	Success - New CSMS Root - New CSMS	M		For CS: at least two configuration slots for networkConnectionProfiles must be supported		

5.1.3. Page 13-23 - (2023-12) - A number of test cases cannot be run for a Charging Station that only supports NoAuthorization as a local authorization option

We found that a number of test cases cannot be run for a Charging Station that only supports NoAuthorization as a local authorization option (and it is not possible with the supported remote authorization options). Test cases for statuses like Invalid, Authorization Cache, Local Auth. List, GroupId etc. will be dropped for this type of Charging Station implementation.

	OCTT Id	OCPP Compliance Testing Tool scenario	Conf. Test for Charging Station	Conf. test for CSMS	Condition / remark	Feature no.	Feature
Old text	TC_C_02	Authorization Invalid/Unknown	C	M	Charging Station: - The Charging Station supports at least one of the following local start authorization options C-30, C-31, C-32, C35 - The Charging Station does NOT have a cable lock that prevents the EV driver to connect the EV and EVSE before authorization.	(C-30 or C-31 or C-32 or C-35) and NOT AQ-2	Local Authorization - using RFID ISO14443 / RFID ISO15693 / KeyCode / NoAuthorization and Does the Charging Station have a cable lock, which prevents the EV driver to connect the EV and EVSE before authorization?
New text	TC_C_02	Authorization Invalid/Unknown	C	M	Charging Station: - The Charging Station supports at least one of the following local start authorization options C-30, C-31, C-32 - The Charging Station does NOT have a cable lock that prevents the EV driver to connect the EV and EVSE before authorization.	(C-30 or C-31 or C-32) and NOT AQ-2	Local Authorization - using RFID ISO14443 / RFID ISO15693 / KeyCode and Does the Charging Station have a cable lock, which prevents the EV driver to connect the EV and EVSE before authorization?
Old text	TC_C_05	Authorization invalid - Cable lock	C		For CS: - The Charging Station has a cable lock, which prevents the EV driver to connect the EV and EVSE before authorization. - The Charging Station supports at least one of the following local start authorization options C-30, C-31, C-32, C35 - The Charging Station does NOT have the following configuration: TxStartPoint ReadOnly AND value Authorized is NOT set.	(C-30 or C-31 or C-32 or C-35) and AQ-2	Local Authorization - using RFID ISO14443 / RFID ISO15693 / KeyCode / NoAuthorization
New text	TC_C_05	Authorization invalid - Cable lock	C		For CS: - The Charging Station has a cable lock, which prevents the EV driver to connect the EV and EVSE before authorization. - The Charging Station supports at least one of the following local start authorization options C-30, C-31, C-32. - The Charging Station does NOT have the following configuration: TxStartPoint ReadOnly AND value Authorized is NOT set.	(C-30 or C-31 or C-32) and AQ-2	Local Authorization - using RFID ISO14443 / RFID ISO15693 / KeyCode
Old text	TC_E_52	DisableRemoteAuthorization	C		If the Charging Station supports the option for disabling remote authorization	C-58	

	OCTT Id	OCPP Compliance Testing Tool scenario	Conf. Test for Charging Station	Conf. test for CSMS	Condition / remark	Feature no.	Feature
New text	TC_E_52	DisableRemoteAuthorization	C		If the Charging Station supports the option for disabling remote authorization and The Charging Station supports at least one of the following local start authorization options C-30, C-31, C-32 and Either Authorization Cache or Local Authorization List is supported.	C-58 and (C-30 or C-31 or C-32) and (C-49 or Local Authorization List Management)	Local Authorization - using RFID ISO14443 / RFID ISO15693 / KeyCode & Authorization Cache & Local Authorization List.
Old text	TC_E_16	Deauthorized - Invalid idToken	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. Charging Station: If one or more of the local start authorization options is implemented. AND either a cache, local authorization list or UnknownIdtag (C15) is implemented.	(C-10.2 or C-10.3) and (C-30 - C-35 or ISO 15118 support) and C-01	Supported Transaction Stop Points & Local Authorization options for local start & Authorization - eMAID
New text	TC_E_16	Deauthorized - Invalid idToken	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. Charging Station: If one or more of the local start authorization options is implemented. AND either a cache, local authorization list or UnknownIdtag (C15) is implemented.	(C-10.2 or C-10.3) and (C-30 - C-32 or ISO 15118 support) and C-01	Supported Transaction Stop Points & Local Authorization options for local start & Authorization - eMAID
Old text	TC_E_43	Transaction during offline period	C		Charging Station: If one or more of the local start authorization options is implemented.	C-01 and (C-30 - C-35 or ISO 15118 support)	Offline transaction support & Local Authorization options for local start
New text	TC_E_43	Transaction during offline period	C		Charging Station: If offline authorization is supported and one or more of the local start authorization options is implemented. Or the Charging Station supports NoAuthorization.	(C-01 and (C-30 - C-34 or ISO 15118 support)) or C-35	Offline transaction support & Local Authorization options for local start or NoAuthorization support
Old text	TC_E_44	Stop transaction during offline period	C		Charging Station: If one or more of the local start authorization options is implemented.	C-01 and (C-30 - C-35 or ISO 15118 support)	Offline transaction support & Local Authorization options for local start & Authorization - eMAID

	OCTT Id	OCPP Compliance Testing Tool scenario	Conf. Test for Charging Station	Conf. test for CSMS	Condition / remark	Feature no.	Feature
New text	TC_E_44	Stop transaction during offline period	C		Charging Station: If one or more of the local start authorization options is implemented.	C-30 - C-35 or ISO 15118 support	Local Authorization options for local start & Authorization - eMAID

5.1.4. Page 19 - (2023-12) - TC_E_20_CS Improved condition / remark and aligned the conditions at feature no.

TC_E_20_CS is the equivalent of TC_E_54_CS, that covers the scenario for a Charging Station that supports charging a IEC 61851-1 EV. The condition / remark was still missing this information. The condition listed at the feature no. column is one of the most complicated ones that exists in part 5. We also noticed that it did not correctly cover the described remarks for all permutations, so it has been improved.

	OCTT Id	OCPP Compliance Testing Tool scenario	Conf. Test for Charging Station	Conf. test for CSMS	Condition / remark	Feature no.	Feature
Old text	TC_E_20	EVDisconnected - EV side (able to charge IEC 61851-1 EV)	C	M	TxStopPoint can either be ReadOnly with a subset of the values or have a valueList of supported values, that contains a subset. This testcase is applicable if the value EVConnected is a supported value. And it should be possible to not set EnergyTransfer and PowerPathClosed AND The Charging Station does NOT have following configuration combination; StopTxOnEVSideDisconnect mutability ReadOnly with value *true* AND TxStopPoint mutability is *ReadOnly* and contains *Authorized*	C-10.1 and (C-52 or NOT (C-10.2 or C-10.3 or C-10.4)) AND NOT C-06.1) AND (AQ-9 OR Product Subtype "Mode 1/2-only Charging Station")	Supported Transaction Stop points

	OCTT Id	OCPP Compliance Testing Tool scenario	Conf. Test for Charging Station	Conf. test for CSMS	Condition / remark	Feature no.	Feature
New text	TC_E_20	EVDisconnected - EV side (able to charge IEC 61851-1 EV)	C	M	TxStopPoint can either be ReadOnly with a subset of the values or have a valueList of supported values, that contains a subset. This testcase is applicable if the value EVConnected is a supported value. And it should be possible to not set EnergyTransfer and PowerPathClosed AND The Charging Station does NOT have following configuration combination; StopTxOnEVSideDisconnect mutability ReadOnly with value *true* AND TxStopPoint mutability is *ReadOnly* and contains *Authorized* AND the Charging Station supports charging an EV using IEC 61851-1 (Mode 3) or is a Mode 1/2-only Charging Station.	(C-10.1 AND (NOT (NOT C-52 AND (C-10.3 or C-10.4))) AND NOT (NOT C.06.1 AND NOT C-52 AND C-10.2)) AND (AQ-9 OR Product Subtype "Mode 1/2-only Charging Station")	Supported Transaction Stop points

5.1.5. Page 20 - (2023-12) - TC_E_54_CS Improved condition / remark and aligned the conditions at feature no.

TC_E_54_CS was created to accommodate testing TC_E_20_CS with a Charging Station that supports high level communication. We noticed that the OCPP communication is different in that case. The condition / remark was still missing this information. The condition listed at the feature no. column is one of the most complicated ones that exists in part 5. We also noticed that it did not correctly cover the described remarks for all permutations, so it has been improved.

	OCTT Id	OCPP Compliance Testing Tool scenario	Conf. Test for Charging Station	Conf. test for CSMS	Condition / remark	Feature no.	Feature
Old text	TC_E_54	EVDisconnected - EV side (not able to charge IEC 61851-1 EV)	C		TxStopPoint can either be ReadOnly with a subset of the values or have a valueList of supported values, that contains a subset. This testcase is applicable if the value EVConnected is a supported value. And it should be possible to not set EnergyTransfer and PowerPathClosed AND The Charging Station does NOT have following configuration combination; StopTxOnEVSideDisconnect mutability ReadOnly with value *true* AND TxStopPoint mutability is *ReadOnly* and contains *Authorized*	C-10.1 and (C-52 or NOT (C-10.2 or C-10.3 or C-10.4)) AND (HFS-4 OR ISO15118 support) AND NOT Product Subtype "Mode 1/2-only Charging Station"	Supported Transaction Stop points

	OCTT Id	OCPP Compliance Testing Tool scenario	Conf. Test for Charging Station	Conf. test for CSMS	Condition / remark	Feature no.	Feature
New text	TC_E_54	EVDisconnected - EV side (DC and/or ISO-15118 Support)	C		TxStopPoint can either be ReadOnly with a subset of the values or have a valueList of supported values, that contains a subset. This testcase is applicable if the value EVConnected is a supported value. And it should be possible to not set EnergyTransfer and PowerPathClosed AND The Charging Station does NOT have following configuration combination; StopTxOnEVSideDisconnect mutability ReadOnly with value *true* AND TxStopPoint mutability is *ReadOnly* and contains *Authorized* AND the Charging Station has DC or ISO-15118 support AND is NOT a Mode 1/2-only Charging Station.	C-10.1 AND (NOT (NOT C-52 AND (C-10.2 or C-10.3 or C-10.4))) AND (HFS-4 OR ISO15118 support) AND NOT Product Subtype "Mode 1/2-only Charging Station"	Supported Transaction Stop points

5.1.6. Page 21 - (2023-12) - TC_E_39_CS - Testcase not only applicable for TxStopPoint Authorized

This test case was conditional for TxStopPoint Authorized, however this testcase can always be performed. Due to the importance of this functionality, the condition has been removed and the testcase has become mandatory.

	OCTT Id	OCPP Compliance Testing Tool scenario	Conf. Test for Charging Station	Conf. test for CSMS	Condition / remark	Feature no.	Feature
Old text	TC_E_39	Deauthorized - timeout	C	M	TxStopPoint can either be ReadOnly with a subset of the values or have a valueList of supported values, that contains a subset. This testcase is applicable if the value Authorized is a supported value.	C-10.2	Supported Transaction Stop points
New text	TC_E_39	Deauthorized - timeout	M	M			

5.1.7. Page 24 - (2023-12) - TC_F_04_CS should only be applicable when TxStartPoint Authorized or ParkingBayOccupancy are supported

For this cable plugin timeout testcase we can only check the transmitted OCPP TransactionEventRequest messages, to validate the behavior. So the testcase is only testable if the Charging Station supports starting the transaction before the cable is plugged in. If the Charging Station does not support TxStartPoint Authorized or ParkingBayOccupancy, it must still support the cable

plugin timeout mechanism itself.

	OCTT Id	OCPP Compliance Testing Tool scenario	Conf. Test for Charging Station	Conf. test for CSMS	Condition / remark	Feature no.	Feature
Old text	TC_F_04	Remote start first - Cable plugin timeout	M	M			
New text	TC_F_04	Remote start first - Cable plugin timeout	C	M	TxStartPoint can either be ReadOnly with a subset of the values or have a valueList of supported values, that contains a subset. This testcase is applicable if the value Authorized or ParkingBayOccupancy is a supported value.	C-09.2 or C-09.6	Supported Transaction Start points

6. Part 6

6.1. Test Cases Charging Station

6.1.1. Page 3 - (2023-12) - General tool rules/validations - Added information for idToken type *NoAuthorization*

Added	When idToken type *NoAuthorization* is configured to be used, the OCTT will act/validate differently. No AuthorizeRequest is expected anymore and the value of the idToken at the TransactionEventRequest should be an empty string "". Additionally many testcases like Authorization cache, local authorization list, groupIdToken, etc. Will not work for this idToken type.

6.1.2. Page 30 - (2023-12) - TC_B_30_CS - Removed prerequisite and added note

The testcase has been made more robust to also work for Charging Station that disconnect during this testcase.

Old text	**Prerequisite(s)**	The Charging Station is configured to keep the connection open while it is waiting to resend the BootNotificationRequest.
New text	**Prerequisite(s)**	

Added note to main step

Old text	**3.** The OCTT sends a **GetBaseReportRequest** with **reportBase** *FullInventory*
New text	**3.** The OCTT sends a **GetBaseReportRequest** with **reportBase** *FullInventory* Note(s) : The OCTT will only send this request if the Charging Station does not disconnect

6.1.3. Page 36 - (2023-12) - TC_B_08_CS - Removed testcase

This test case tests requirement B06.FR.05, which is not a Charging Station requirement. The limit must be respected, not tested by the OCTT. There fore it will be removed from part 6.

Removed testcase

6.1.4. Page 42 - (2023-12) - TC_B_11_CS - Changed hardcoded values for integer and decimal to configurable values

The defined hardcoded values were not usable for all Charging Stations.

Changed main step

Old text	Notes: *Steps 1 to 8 are repeated 5 times for value = 1, 1.1, true, currentTime, "abc"*
New text	Notes: *Steps 1 to 8 are repeated 5 times for value = <configured offlineThreshold>, <configured offlineThreshold + 0.1>, true, currentTime, "abc"*

Additionally, step 3 and 4 regarding setting values to "SmartChargingCtrlr.LimitChangeSignificance" is only tested if the Charging Station supports it.

Added note to main steps

Added	Notes: *Steps 3 and 4 will only be tested if this component/variable combination is supported*

6.1.5. Page 50 - (2023-12) - TC_B_21_CS - Removed requirement reference

This requirement was removed from part 2 specification.

Old text	**Requirement(s)**	B12.FR.01, B12.FR.03 , E07.FR.03
New text	**Requirement(s)**	B12.FR.01, B12.FR.03

6.1.6. Page 56 - (2023-12) - TC_B_41_CS - Typo step reference

Typo

Changed main step

Old text	**8.** Execute **Reusable State** *EVConnectedPostSession* for EVSE.id = 2 Note(s): If TxStopPoint contains one of the following values; Authorized, EnergyTransfer, PowerPathClosed, DataSigned. Then the transaction will have ended at the *EVConnectedPostSession* state AND the Charging Station will proceed with resetting itself. Proceed to step 10 Else proceed with step 9.
New text	**8.** Execute **Reusable State** *EVConnectedPostSession* for EVSE.id = 2 Note(s): If TxStopPoint contains one of the following values; Authorized, EnergyTransfer, PowerPathClosed, DataSigned. Then the transaction will have ended at the *EVConnectedPostSession* state AND the Charging Station will proceed with resetting itself. Proceed to step 11 Else proceed with step 9.

6.1.7. Page 59 - (2023-12) - TC_B_26_CS - Removed rebooting step

The Charging Station does not reboot, when a reset EVSE is requested by the CSMS.

Removed main step

Removed	**7.** ChargingStation Reboots

Additionally requirement E07.FR.03 was removed from part 2 specification.

Old text	**Requirement(s)**	B12.FR.01, B12.FR.03 , E07.FR.03
New text	**Requirement(s)**	B12.FR.01, B12.FR.03

6.1.8. Page 64/66 - (2023-12) - TC_B_45_CS & TC_B_46_CS - Testcase has been made more robust for Charging Stations that do not automatically reboot.

The testcase has been made more robust for Charging Stations that respond with Accepted, but do not automatically reboot.

Changed main step

Old text	**5.** The OCTT sends a **ResetRequest** with **type** *OnIdle* Note(s): *- This step will only be executed when the status RebootRequired is returned at step 4.*

New text	**5.** The OCTT sends a **ResetRequest** with **type** *OnIdle* Note(s): *- This step will only be executed when the status RebootRequired is returned at step 4 , or if the charging does not automatically reboot.*

6.1.9. Page 68-72 - (2023-12) - TC_B_45_CS-TC_B_50_CS - Resolved testcase inconsistency regarding used configuration slots

The testcase dynamically uses either configuration slot 1 or 2, based on the one that is currently connected. So the configurationSlot and NetworkConfigurationPriority also needs to be set dynamically.

Changed main step

Old text	**1.** The OCTT sends a **SetNetworkProfileRequest** with **configurationSlot** is *<Configured configurationSlot>* or *<Configured configurationSlot>* depending on which one is already in use - **connectionData.messageTimeout** *<Configured messageTimeout2>* - **connectionData.ocppCsmsUrl** *<ocppCsmsUrl that is not currently active>* - **connectionData.ocppInterface** *<Configured ocppInterface2>* - **connectionData.ocppVersion** *OCPP20* - **connectionData.securityProfile** *<Configured securityProfile2>*
New text	**1.** The OCTT sends a **SetNetworkProfileRequest** with **configurationSlot** is *<Configured configurationSlot>* or *<Configured configurationSlot 2>* depending on which one is already in use - **connectionData.messageTimeout** *<Configured messageTimeout>* or *<Configured messageTimeout2>* - **connectionData.ocppCsmsUrl** *<ocppCsmsUrl that is not currently active>* - **connectionData.ocppInterface** *<Configured ocppInterface>* or *<Configured ocppInterface2>* - **connectionData.ocppVersion** *OCPP20* - **connectionData.securityProfile** *<Configured securityProfile2>* or *<Configured securityProfile2>*

Changed main step

Old text	**3.** The OCTT sends a **SetVariablesRequest** with **variable.name** is *"NetworkConfigurationPriority"* **component.name** is *"OCPPCommCtrlr"* **attributeValue** is *<Configured configurationSlot2>*
New text	**3.** The OCTT sends a **SetVariablesRequest** with **variable.name** is *"NetworkConfigurationPriority"* **component.name** is *"OCPPCommCtrlr"* **attributeValue** is *Configured slot from Step 1, the previously configured slot*

6.1.10. Page 72 - (2023-12) - TC_B_50_CS - Testcase not only applicable for AdditionalRootCertificateCheck = true

This test case was conditional for feature AdditionalRootCertificateCheck, however this can always be performed (no relation with AdditionalRootCertificateCheck).

Old text	**Prerequisite(s)**	- The Charging Station supports AS-2: AdditionalRootCertificateCheck. - Configured (new) CSMS Root certificate 2 must be signed by the configured (old) CSMS Root certificate 2. - At least two configuration slots for networkConnectionProfiles must be supported
New text	**Prerequisite(s)**	At least two configuration slots for networkConnectionProfiles must be supported

Removed main steps

Removed	**10.** The OCTT sends a **GetInstalledCertificateIdsRequest** with **certificateType** is *CSMSRootCertificate* **11.** The Charging Station responds with a **GetInstalledCertificateIdsResponse**

Removed tool validation

Old text	* Step 6: Message **ResetResponse** - **status** *Accepted* ***** Step 11: Message: **GetInstalledCertificateIdsResponse** - **status** must be *Accepted* - **certificateHashDataChain** must NOT contain an entry with following values: - **certificateType** is *CSMSRootCertificate* - **certificateHashData** contains *<HashData from configured old CSMS Root certificate>* *NOTE:* The Charging Station dropped the (old) fallback certificate, because it was able to connect using the (new) Root certificate.
New text	* Step 6: Message **ResetResponse** - **status** *Accepted*

6.1.11. Page 77 - (2023-12) - TC_B_53_CS - Removed Component / variable list

It does not make sense to create a duplication of the component / variable list that is already defined in part 2 specification. This will only increase the chance of inconsistencies.

Removed component / variable table

Changed post scenario validation

Old text	OCTT checks that at least the following variables are reported:
New text	The OCTT checks that the components / variables that are required according to the OCPP specification are implemented.

6.1.12. Page 82-99 - (2023-12) - TC_C_02_CS-TC_C_57_CS - A number of test cases cannot be run for a Charging Station that only supports NoAuthorization as a local authorization option

We found that a number of test cases cannot be run for a Charging Station that only supports NoAuthorization as a local authorization option (and it is not possible with the supported remote authorization options). Test cases for statuses like Invalid, Authorization Cache, Local Auth. List, GroupId etc. will be dropped for this type of Charging Station implementation.

This erratum is applicable for the following testcases; TC_C_02_CS, TC_C_05_CS, TC_C_06_CS, TC_C_07_CS, TC_C_09_CS, TC_C_10_CS, TC_C_11_CS, TC_C_34_CS, TC_C_36_CS, TC_C_39_CS, TC_C_44_CS, TC_C_45_CS, TC_C_47_CS, TC_C_48_CS, TC_C_49_CS, TC_C_56_CS, TC_C_57_CS, TC_E_16_CS, TC_E_52_CS

Added	**Prerequisite(s)**	The Charging Station supports authorization methods other than NoAuthorization

6.1.13. Page 93 - (2023-12) - TC_C_15_CS - Improvements based on experience from additional testing

During testing it was noticed that a value of 500 for **MaxEnergyOnInvalidId** is not enough. The scope of this testcase is to test that the Charging Station does **not** deathorize the transaction.

Changed preparations

Old text	**Configuration State:**	**AuthCacheCtrlr.AuthCacheEnabled** is *true* (If implemented) **AuthCtrlr.LocalPreAuthorize** is *true* (If implemented) **AuthCtrlr.LocalAuthorizeOffline** is *true* **OfflineTxForUnknownIdEnabled** is *true* (If implemented) **StopTxOnInvalidId** is *false* **MaxEnergyOnInvalidId** is *500* **OfflineThreshold** is *<Configured RetryBackOffWaitMinimum_duration> + 60.0* **RetryBackOffWaitMinimum** is *<Configured RetryBackOffWaitMinimum_duration>* **RetryBackOffRandomRange** is *0* Note: *<Configured RetryBackOffWaitMinimum_duration should be long enough to execute manual tasks>*
New text	**Configuration State:**	**AuthCacheCtrlr.AuthCacheEnabled** is *true* (If implemented) **AuthCtrlr.LocalPreAuthorize** is *true* (If implemented) **AuthCtrlr.LocalAuthorizeOffline** is *true* **OfflineTxForUnknownIdEnabled** is *true* (If implemented) **StopTxOnInvalidId** is *false* **MaxEnergyOnInvalidId** is *10.000* **OfflineThreshold** is *<Configured RetryBackOffWaitMinimum_duration> + 60.0* **RetryBackOffWaitMinimum** is *<Configured RetryBackOffWaitMinimum_duration>* **RetryBackOffRandomRange** is *0* Note: *<Configured RetryBackOffWaitMinimum_duration should be long enough to execute manual tasks>*

Removed confusing main step. It was intended to describe the Charging Station shall not deathorize the transaction, as mentioned by the tool validation section. But it is more clear to remove the steps as a whole from the main steps.

Removed main steps

Removed	**5.** The Charging Station sends a **TransactionEventRequest** with **triggerReason** *Deauthorized* **6.** The OCTT responds with a **TransactionEventResponse**

It is also not allowed for the Charging Station to stop the energy transfer.

Changed tool validation

Old text	* Step 2: Message **TransactionEventRequest** A message with (optional): - **triggerReason** *Authorized* - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>* - **offline** *True* A message with: - **triggerReason** *ChargingStateChanged* - **offline** *True* No message with: - **triggerReason** *Deauthorized* or - **transactionInfo.chargingState** *SuspendedEVSE*

New text	* Step 3: Message **TransactionEventRequest** A message with (optional): - **triggerReason** *Authorized* - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>* - **offline** *True* A message with: - **triggerReason** *ChargingStateChanged* - **offline** *True* No message with: - **triggerReason** *Deauthorized* or - **triggerReason** *ChargingStateChanged* and - **transactionInfo.chargingState** *SuspendedEVSE*

6.1.14. Page 101 - (2023-12) - TC_C_33_CS - Fixed broken table

The tool validations dropped of, because there was an Asciidoc issue. The table has been restored.

6.1.15. Page 104 - (2023-12) - TC_C_37_CS - Editorial issue

triggerReason *Authorized* should have been part of **TransactionEventRequest**, not **TransactionEventResponse**.

Changed main step

Old text	**7.** The Charging Station sends an **TransactionEventRequest** **8.** The OCTT responds with an **TransactionEventResponse** with **triggerReason** *Authorized*
New text	**7.** The Charging Station sends an **TransactionEventRequest** with **triggerReason** *Authorized* **8.** The OCTT responds with an **TransactionEventResponse** with

6.1.16. Page 131 - (2023-12) - TC_E_39_CS - Removed (local) indication on Authorized reusable state

This testcase is also possible for remote authorization.

Changed preparations

Old text	**Reusable State(s):**	**State** is *Authorized* (local)
New text	**Reusable State(s):**	**State** is *Authorized*

6.1.17. Page 131 - (2023-12) - TC_E_39_CS - Made testcase more flexible to handle all TxStart/StopPoint combinations

The testcase was only able to handle TxStart/StopPoint Authorized, but has been improved to also able to handle all other TxStart/StopPoint combinations.

Changed main steps

Old text	**1.** The Charging Station sends a **TransactionEventRequest** Note(s): *- This step needs to be executed after the <Configured ev_connection_timeout> expires, if the transaction has been started. So in the case **TxStartPoint** contains ParkingBayOccupancy OR Authorized* **2.** The OCTT responds with a **TransactionEventResponse** Note(s) : *Optionally the Charging Station can send a StatusNotificationRequest or NotifyEventRequest with status Available*
New text	**1.** The Charging Station sends a **TransactionEventRequest** Note(s): *- This step needs to be executed after the <Configured ev_connection_timeout> expires, if the transaction has been started. So in the case **TxStartPoint** contains ParkingBayOccupancy OR Authorized* **2.** The OCTT responds with a **TransactionEventResponse** Note(s) : *Step 1 and 2 are optional and will only be expected when the TxStartPoint is set to ParkingBayOccupancy or Authorized.* *Optionally the Charging Station can send a StatusNotificationRequest or NotifyEventRequest with status Available.* Manual Action : *Connect the EV and EVSE on EV side.* Manual Action : *Connect the EV and EVSE on EVSE side.* **3.** The Charging Station sends a **TransactionEventRequest** Note(s) : *- This step needs to be executed after the <Configured ev_connection_timeout> expires, if the transaction has been started. So in the case **TxStartPoint** contains ParkingBayOccupancy OR Authorized* **4.** The OCTT responds with a **TransactionEventResponse** Note(s) : *Charging Station is allowed to sent a TransactionEventRequest for the cableplugin event when this is applicable, but should not start charging.*

Changed tool validation

Old text	* Step 1: Message: **TransactionEventRequest** - **triggerReason** must be *EVConnectTimeout* - **eventType** must be *Ended* - **transactionInfo.stoppedReason** must be *Timeout*
New text	* 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: **TransactionEventRequest** - **triggerReason** can only be *CablePluggedIn* - **transactionInfo.chagringState** should not be *Charging* - **eventType** must be *Updated* if TxStartPoint is *ParkingBayOccupancy* , else *Ended*

6.1.18. Page 143 - (2023-12) - TC_E_14_CS - Explicitly describe it is allowed to omit the stoppedReason in case of Local

The OCTT already allowed omitting the stoppedReason in case of Local as described by the specification, but part 6 did not explicitly describe this.

Changed tool validation

Old text	* Step 3: Message: **TransactionEventRequest** - **triggerReason** must be *EVCommunicationLost* - **transactionInfo.chargingState** must be *Idle* - If the OCTT is configured to stop transactions using a RequestStopTransactionRequest message then **transactionInfo.stoppedReason** must be *Remote* Else **transactionInfo.stoppedReason** must be *Local* or *EVDisconnected* - **eventType** must be *Ended*
New text	* Step 3: Message: **TransactionEventRequest** - **triggerReason** must be *EVCommunicationLost* - **transactionInfo.chargingState** must be *Idle* - If the OCTT is configured to stop transactions using a RequestStopTransactionRequest message then **transactionInfo.stoppedReason** must be *Remote* Else **transactionInfo.stoppedReason** must be *Local*, *EVDisconnected* or be omitted. - **eventType** must be *Ended*

6.1.19. Page 158 - (2023-12) - TC_E_31_CS - Made testcase more robust and flexible regarding local / remote start/stop

The OCTT and testcases should be flexible. So it is now possible to run this testcase with a Charging Station that only supports remote start/stop, however under very specific circumstances it is not possible to run this testcase with remote start/stop, as described by below adjusted prerequisites.

Old text	**Prerequisite(s)**	The Charging Station supports at least one authorization method described at the following Use cases; C01, C02, C04.
New text	**Prerequisite(s)**	The Charging Station supports at least one authorization method described at the following Use cases; C01, C02, C04 and and the following configuration is not present: - <configured scenario> is remote and - TxStopPoint is Authorized and - TxCtrlr.StopTxOnEVSideDisconnect is not true and cannot be configured that way.

The testcase has been made more robust. It now uses and takes into account the OCTT configuration *Transaction duration*. Additionally the Sampled meter values are enabled to increase the chances of there being a TransactionEventRequest message in the queue. With only the TransactionEventRequest with eventType = Ended in the queue, the Charging Station might empty its queue a fraction of a second, before the OCTT is able to send the GetTransactionStatusRequest.

Changed preparations

Old text	**Configuration State:**	**OfflineThreshold** is *<Configured RetryBackOffWaitMinimum_duration> + 60.0* **RetryBackOffWaitMinimum** is *<Configured RetryBackOffWaitMinimum_duration>* **RetryBackOffRandomRange** is *0* Note: *<Configured RetryBackOffWaitMinimum_duration> should be long enough to execute manual tasks after waiting for <Configured Transaction Duration> seconds*

New text	**Configuration State:**	**SampledDataTxUpdatedMeasurands** is *<Configured transaction_updated_metervalues_measurands>* **SampledDataTxUpdatedInterval** is *<Configured transaction_updated_metervalues_interval>* **OfflineThreshold** is *<Configured RetryBackOffWaitMinimum_duration> + <Configured Transaction Duration> + 60.0* **RetryBackOffWaitMinimum** is *<Configured RetryBackOffWaitMinimum_duration> + <Configured Transaction Duration>* **RetryBackOffRandomRange** is *0* Note: *<Configured Transaction Duration> should be long enough to execute manual tasks*

The manual action to present the same idToken as used to start the transaction is only required if the OCTT is configured to run the testcase in local authorization mode.

Added note to main step

Added	Notes(s): *Only if configured scenario is local*

6.1.20. Page 166/167 - (2023-12) - TC_E_42_CS & TC_E_51_CS - Refined the tool validation of the testcase

No matter how high the configured **MessageAttemptsTransactionEvent** is, the OCTT will now this testcase passed after receiving the second message. Another testcase will test the max retry count.

Changed tool validation

Old text	* Step 5: - Needs to be send a number of times equal to *<Configured message_attempts_transaction_event>* with an interval of (*<Configured message_attempts_transaction_event_interval>* * the number of preceding transmissions of this same message) + *OCPPCommCtrlr.MessageTimeout.Default*. - The OCTT waits an additional MessageAttemptsTransactionEvent iteration where the interval is multiplied again, to validate if the Charging Station stops resending the TransactionRequest message(s).
New text	* Step 5: - Needs to be sent 2 times with an interval of (*<Configured message_attempts_transaction_event_interval>* * the number of preceding transmissions of this same message) + *OCPPCommCtrlr.MessageTimeout.Default*. - The OCTT waits an additional MessageAttemptsTransactionEvent iteration where the interval is multiplied again, to validate if the Charging Station stops resending the TransactionRequest message(s).

6.1.21. Page 174 - (2023-12) - TC_F_04_CS - Missing prerequisite

This testcase is only applicable if the Charging Station supports either TxStartPoint Authorized or ParkingBayOccupancy. Otherwise the Charging Station will not have started a transaction. So in that case the OCTT won't be able to verify the TransactionEventRequest with eventType Ended.

Old text	**Prerequisite(s)**	N/a
New text	**Prerequisite(s)**	The Charging Station supports **TxCtrlr.TxStartPoint** *ParkingBayOccupancy* OR *Authorized* .

6.1.22. Page 207 - (2023-12) - TC_G_13_CS - Charging Station does not have to report the status of the connector

The availability is being set from Inoperative to Inoperative, therefore it is not needed for the Charging Station to report the status of the connector to the CSMS, because there was no status change.

Changed main steps

Old text	**3.** The Charging Station notifies the CSMS about the current state of all connectors. **4.** The OCTT responds accordingly.

New text	Note : *It is not needed for the Charging Station to report the status of the connector to the CSMS, because there was no status change.*

Changed tool validation

Old text	* Step 2: Message **ChangeAvailabilityResponse** - **status** *Accepted* * Step 3: Message: **StatusNotificationRequest** - **connectorStatus** *Unavailable* Message: **NotifyEventRequest** - **eventData[0].trigger** *Delta* - **eventData[0].actualValue** *"Unavailable"* - **eventData[0].component.name** *"ChargingStation"* - **eventData[0].variable.name** *"AvailabilityState"*
New text	* Step 2: Message **ChangeAvailabilityResponse** - **status** *Accepted*

6.1.23. Page 217/219/223 - (2023-12) - TC_J_01_CS & TC_J_02_CS & TC_J_06_CS - It is currently not possible to send a NotifyEventRequest instead of a MeterValuesRequest

Part 2 specification describes that the **StatusNotificationRequest** and the **MeterValuesRequest** will be moved to the device model at some point and it can be transmitted via a NotifyEventRequest. As for the **StatusNotificationRequest**, this is already described how to do this, so the OCTT will expect a Charging Station to send NotifyEventRequest messages instead. However as for the **MeterValuesRequest**, it was noticed that this is not clearly described. The different measurands need to transmitted using several different components and variables. Therefore it is not allowed to send these instead of the **MeterValuesRequest** messages, however it is allowed to send them in parallel.

Changed tool validation

Old text	* Step 1: Message: **MeterValuesRequest** - **sampledValue[0].context** must be *Sample.Clock* - **sampledValue** must contain *<An element per configured measurand at the AlignedDataMeasurands. The measurand field may be omitted when the measurand is "Energy.Active.Import.Register">* Message: **NotifyEventRequest** - **eventData** must contain *<An element per configured measurand at the AlignedDataMeasurands.>* - **trigger** must be *Periodic* - **component.name** must be *"FiscalMetering"* *Note: The following tool validation will NOT be validated by the OCTT:* - **variable.name** must *<Refer to the configured measurand in PascalCase without a "." in between. For example; "EnergyActiveImportRegister">*
New text	* Step 1: Message: **MeterValuesRequest** - **sampledValue[0].context** must be *Sample.Clock* - **sampledValue** must contain *<An element per configured measurand at the AlignedDataMeasurands. The measurand field may be omitted when the measurand is "Energy.Active.Import.Register">* *Note: The following tool validation will NOT be validated by the OCTT:* - **variable.name** must *<Refer to the configured measurand in PascalCase without a "." in between. For example; "EnergyActiveImportRegister">*

Changed post scenario validation

Old text	Message: **MeterValuesRequest** - **timestamp** *<The intervals between the timestamps of the received Meter Value messages must equal the configured value at AlignedDataInterval. However it is allowed to send multiple Meter Value messages per configured interval. One (or more in case the amount of measured data is too much for one message) for each EVSE and one (or more) for the main power meter (evseId=0). But the timestamp of these messages must all be the same.>* Message: **NotifyEventRequest** - **timestamp** *<The intervals between the timestamps of the received Meter Value messages must equal the configured value at AlignedDataInterval. However it is allowed to send multiple Meter Value messages per configured interval. One (or more in case the amount of measured data is too much for one message) for each EVSE and one (or more) for the main power meter (evseId=0). But the timestamp of these messages must all be the same.>*
New text	Message: **MeterValuesRequest** - **timestamp** *<The intervals between the timestamps of the received Meter Value messages must equal the configured value at AlignedDataInterval. However it is allowed to send multiple Meter Value messages per configured interval. One (or more in case the amount of measured data is too much for one message) for each EVSE and one (or more) for the main power meter (evseId=0). But the timestamp of these messages must all be the same.>*

6.1.24. Page 232-265 - (2023-12) - TC_L_XX_CS - Update testcase structure L group testcases

The structure of almost all 'Secure Firmware Update' testcases have been updated. There were several reasons for this. Please note that this has mostly been done for readability. Functional changes that have been made, were mostly to increase the flexibility needed to comply with the requirements and all possible firmware update process paths defined at part 2. Please refer to Part 2 specification Figure 119. Firmware update process, for the overview.

Table 4. Test Case Id: TC_L_01_CS

Test case name	**Secure Firmware Update - Installation successful**	
Test case Id	TC_L_01_CS	
Use case Id(s)	L01	
Requirement(s)	L01.FR.01,L01.FR.04,L01.FR.05,L01.FR.09,L01.FR.10,L01.FR.12,L01.FR.13,L01.FR.15,L01.FR.20,L01.FR.21,L01.FR.23	
System under test	Charging Station	
Description	The CSMS is able to request the Charging Station to securely download and install a new firmware by sending an UpdateFirmwareRequest with a signingCertificate.	
Purpose	To verify if the Charging Station is able to securely download and install a new firmware.	
Prerequisite(s)	A file server has been setup according to the (by the Charging Station) supported file transfer protocol(s), indicated by the configuration variable **FileTransferProtocols**.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **UpdateFirmwareResponse**	**1.** The OCTT sends a **UpdateFirmwareRequest** with **firmware.installDateTime** *<Current DateTime - 2 hours>* **firmware.location** *<Configured firmware_location>* **firmware.retrieveDateTime** *<Current DateTime - 2 hours>* **firmware.signingCertificate** *<Configured signingCertificate>* **firmware.signature** *<Configured signature>*
	3. The Charging Station sends a **FirmwareStatusNotificationRequest** With **status** *Downloading*	**4.** The OCTT responds with a **FirmwareStatusNotificationResponse**
	5. The Charging Station sends a **FirmwareStatusNotificationRequest** With **status** *Downloaded*	**6.** The OCTT responds with a **FirmwareStatusNotificationResponse**
	7. The Charging Station sends a **FirmwareStatusNotificationRequest** With **status** *SignatureVerified*	**8.** The OCTT responds with a **FirmwareStatusNotificationResponse**
	9. The Charging Station notifies the CSMS about the current state of all connectors. Note(s): *- This step is optional. The Charging Station may wants to set its connectors to Unavailable, before proceeding installing the new firmware.*	**10.** The OCTT responds accordingly.

Test case name	Secure Firmware Update - Installation successful	
	11. Execute **Reusable State** *RebootBeforeFirmwareInstallation* Note: *This step only needs to be executed if the Charging Station needs to reboot before firmware installation.*	
	12. The Charging Station sends a **FirmwareStatusNotificationRequest** With **status** *Installing* Note(s): *- This step only needs to be executed if the Charging Station did NOT reboot before firmware installation, at step 11.*	**13.** The OCTT responds with a **FirmwareStatusNotificationResponse**
	14. Execute **Reusable State** *RebootBeforeFirmwareActivation* Note: *This step only needs to be executed if the Charging Station needs to reboot before firmware activation.*	
	15. The OCTT waits for the Charging Station to reconnect. Note: *This step only needs to be executed if the Charging Station did not reboot/reconnect up until this point. The Charging Station should at least reconnect to reestablish the protocol version handshake.*	
	Note: *Step 16 through 21 can be send in a different order.*	
	16. The Charging Station notifies the CSMS about the current state of all connectors. Note(s): *- This step only needs to be executed if the connectors were previously set to Unavailable (at step 9) and the Charging Station did not report setting them back to Available (after a reboot sequence at step 11 or 14) yet.*	**17.** The OCTT responds accordingly.
	18. The Charging Station sends a **FirmwareStatusNotificationRequest** With **status** *Installed*	**19.** The OCTT responds with a **FirmwareStatusNotificationResponse**
	20. The Charging Station sends a **SecurityEventNotificationRequest** With **type** *FirmwareUpdated*	**21.** The OCTT responds with a **SecurityEventNotificationResponse**

Test case name	Secure Firmware Update - Installation successful
Tool validations	* Step 2: Message **UpdateFirmwareResponse** - **status** *Accepted* * Step 3: Message **FirmwareStatusNotificationRequest** - **status** *Downloading* * Step 5: Message **FirmwareStatusNotificationRequest** - **status** *Downloaded* * Step 7: Message **FirmwareStatusNotificationRequest** - **status** *SignatureVerified* * Step 9: Message: **StatusNotificationRequest** - **connectorStatus** *Unavailable* *Or* Message: **NotifyEventRequest** - **eventData[0].trigger** *Delta* - **eventData[0].actualValue** *"Unavailable"* - **eventData[0].component.name** *"Connector"* - **eventData[0].variable.name** *"AvailabilityState"* * Step 12: Message **FirmwareStatusNotificationRequest** - **status** *Installing* * Step 16: Message: **StatusNotificationRequest** - **connectorStatus** *Available* *Or* Message: **NotifyEventRequest** - **eventData[0].trigger** *Delta* - **eventData[0].actualValue** *"Available"* - **eventData[0].component.name** *"Connector"* - **eventData[0].variable.name** *"AvailabilityState"* * Step 18: Message **FirmwareStatusNotificationRequest** - **status** *Installed* * Step 20: Message **SecurityEventNotificationRequest** - **type** *FirmwareUpdated*
	Post scenario validations: N/a

Table 5. Test Case Id: TC_L_02_CS

Test case name	**Secure Firmware Update - InstallScheduled**	
Test case Id	TC_L_02_CS	
Use case Id(s)	L01	
Requirement(s)	L01.FR.01,L01.FR.04,L01.FR.05,L01.FR.09,L01.FR.10,L01.FR.12,L01.FR.15,L01.FR.16,L01.FR.20,L01.FR.21,L01.FR.23	
System under test	Charging Station	
Description	The CSMS is able to request the Charging Station to securely download and install a new firmware by sending an UpdateFirmwareRequest with a signingCertificate.	
Purpose	To verify if the Charging Station is able securely download a new firmware and schedule its installation.	
Prerequisite(s)	- A file server has been setup according to the (by the Charging Station) supported file transfer protocol(s), indicated by the configuration variable **FileTransferProtocols**. - The OCTT configuration firmware installDateTime needs to set to a future dateTime.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **UpdateFirmwareResponse**	**1.** The OCTT sends a **UpdateFirmwareRequest** with **firmware.location** *<Configured firmware_location>* **firmware.retrieveDateTime** *<Current DateTime - 2 hours>* **firmware.signingCertificate** *<Configured signingCertificate>* **firmware.signature** *<Configured signature>* **firmware.installDateTime** *<Current DateTime + <Configured Install Offset Period>>*
	3. The Charging Station sends a **FirmwareStatusNotificationRequest** With **status** *Downloading*	**4.** The OCTT responds with a **FirmwareStatusNotificationResponse**
	5. The Charging Station sends a **FirmwareStatusNotificationRequest** With **status** *Downloaded*	**6.** The OCTT responds with a **FirmwareStatusNotificationResponse**
	7. The Charging Station sends a **FirmwareStatusNotificationRequest** With **status** *SignatureVerified*	**8.** The OCTT responds with a **FirmwareStatusNotificationResponse**
	9. The Charging Station sends a **FirmwareStatusNotificationRequest** With **status** *InstallScheduled* Note(s): *- The Charging Station will start installing the firmware after the set installDateTime is reached.*	**10.** The OCTT responds with a **FirmwareStatusNotificationResponse**
	11. The Charging Station notifies the CSMS about the current state of all connectors. Note(s): *- This step is optional. The Charging Station may wants to set its connectors to Unavailable, before proceeding installing the new firmware.*	**12.** The OCTT responds accordingly.

Test case name	Secure Firmware Update - InstallScheduled	
	13. Execute **Reusable State** *RebootBeforeFirmwareInstallation* Note: *This step only needs to be executed if the Charging Station needs to reboot before firmware installation.*	
	14. The Charging Station sends a **FirmwareStatusNotificationRequest** With **status** *Installing* Note(s): *- This step only needs to be executed if the Charging Station did NOT reboot before firmware installation, at step 13.*	**15.** The OCTT responds with a **FirmwareStatusNotificationResponse**
	16. Execute **Reusable State** *RebootBeforeFirmwareActivation* Note: *This step only needs to be executed if the Charging Station needs to reboot before firmware activation.*	
	17. The OCTT waits for the Charging Station to reconnect. Note: *This step only needs to be executed if the Charging Station did not reboot/reconnect up until this point. The Charging Station should at least reconnect to reestablish the protocol version handshake.*	
	Note: *Step 18 through 23 can be send in a different order.*	
	18. The Charging Station notifies the CSMS about the current state of all connectors. Note(s): *- This step only needs to be executed if the connectors were previously set to Unavailable (at step 11) and the Charging Station did not report setting them back to Available (after a reboot sequence at step 13 or 16) yet.*	**19.** The OCTT responds accordingly.
	20. The Charging Station sends a **FirmwareStatusNotificationRequest** With **status** *Installed*	**21.** The OCTT responds with a **FirmwareStatusNotificationResponse**
	22. The Charging Station sends a **SecurityEventNotificationRequest** With **type** *FirmwareUpdated*	**23.** The OCTT responds with a **SecurityEventNotificationResponse**

Test case name	Secure Firmware Update - InstallScheduled
Tool validations	* Step 2: Message **UpdateFirmwareResponse** - **status** *Accepted* * Step 3: Message **FirmwareStatusNotificationRequest** - **status** *Downloading* * Step 5: Message **FirmwareStatusNotificationRequest** - **status** *Downloaded* * Step 7: Message **FirmwareStatusNotificationRequest** - **status** *SignatureVerified* * Step 9: Message **FirmwareStatusNotificationRequest** - **status** *InstallScheduled* * Step 11: Message: **StatusNotificationRequest** - **connectorStatus** *Unavailable* *Or* Message: **NotifyEventRequest** - **eventData[0].trigger** *Delta* - **eventData[0].actualValue** *"Unavailable"* - **eventData[0].component.name** *"Connector"* - **eventData[0].variable.name** *"AvailabilityState"* * Step 14: Message **FirmwareStatusNotificationRequest** - **status** *Installing* * Step 18: Message: **StatusNotificationRequest** - **connectorStatus** *Available* *Or* Message: **NotifyEventRequest** - **eventData[0].trigger** *Delta* - **eventData[0].actualValue** *"Available"* - **eventData[0].component.name** *"Connector"* - **eventData[0].variable.name** *"AvailabilityState"* * Step 20: Message **FirmwareStatusNotificationRequest** - **status** *Installed* * Step 22: Message **SecurityEventNotificationRequest** - **type** *FirmwareUpdated*
	Post scenario validations: N/a

Table 6. Test Case Id: TC_L_03_CS

Test case name	**Secure Firmware Update - DownloadScheduled**	
Test case Id	TC_L_03_CS	
Use case Id(s)	L01	
Requirement(s)	L01.FR.01,L01.FR.04,L01.FR.05,L01.FR.09,L01.FR.10,L01.FR.12,L01.FR.13,L01.FR.15,L01.FR.20,L01.FR.21,L01.FR.23	
System under test	Charging Station	
Description	The CSMS is able to request the Charging Station to securely download and install a new firmware by sending an UpdateFirmwareRequest with a signingCertificate.	
Purpose	To verify if the Charging Station is able to schedule securely downloading a new firmware.	
Prerequisite(s)	- A file server has been setup according to the (by the Charging Station) supported file transfer protocol(s), indicated by the configuration variable **FileTransferProtocols**. - The OCTT configuration firmware retrieveDateTime needs to set to a future dateTime.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **UpdateFirmwareResponse**	**1.** The OCTT sends a **UpdateFirmwareRequest** with **firmware.installDateTime** *<Current DateTime - 2 hours>* **firmware.location** *<Configured firmware_location>* **firmware.retrieveDateTime** *<Current DateTime + <Configured Download Offset Period>>* **firmware.signingCertificate** *<Configured signingCertificate>* **firmware.signature** *<Configured signature>*
	3. The Charging Station sends a **FirmwareStatusNotificationRequest** With **status** *DownloadScheduled* Note(s): *- The Charging Station will start downloading the firmware after the set retrieveDateTime is reached.*	**4.** The OCTT responds with a **FirmwareStatusNotificationResponse**
	5. The Charging Station sends a **FirmwareStatusNotificationRequest** With **status** *Downloading*	**6.** The OCTT responds with a **FirmwareStatusNotificationResponse**
	7. The Charging Station sends a **FirmwareStatusNotificationRequest** With **status** *Downloaded*	**8.** The OCTT responds with a **FirmwareStatusNotificationResponse**
	9. The Charging Station sends a **FirmwareStatusNotificationRequest** With **status** *SignatureVerified*	**10.** The OCTT responds with a **FirmwareStatusNotificationResponse**
	11. The Charging Station notifies the CSMS about the current state of all connectors. Note(s): *- This step is optional. The Charging Station may wants to set its connectors to Unavailable, before proceeding installing the new firmware.*	**12.** The OCTT responds accordingly.

Test case name	Secure Firmware Update - DownloadScheduled	
	13. Execute **Reusable State** *RebootBeforeFirmwareInstallation* Note: *This step only needs to be executed if the Charging Station needs to reboot before firmware installation.*	
	14. The Charging Station sends a **FirmwareStatusNotificationRequest** With **status** *Installing* Note(s): *- This step only needs to be executed if the Charging Station did NOT reboot before firmware installation, at step 13.*	**15.** The OCTT responds with a **FirmwareStatusNotificationResponse**
	16. Execute **Reusable State** *RebootBeforeFirmwareActivation* Note: *This step only needs to be executed if the Charging Station needs to reboot before firmware activation.*	
	17. The OCTT waits for the Charging Station to reconnect. Note: *This step only needs to be executed if the Charging Station did not reboot/reconnect up until this point. The Charging Station should at least reconnect to reestablish the protocol version handshake.*	
	Note: *Step 18 through 23 can be send in a different order.*	
	18. The Charging Station notifies the CSMS about the current state of all connectors. Note(s): *- This step only needs to be executed if the connectors were previously set to Unavailable (at step 11) and the Charging Station did not report setting them back to Available (after a reboot sequence at step 13 or 16) yet.*	**19.** The OCTT responds accordingly.
	20. The Charging Station sends a **FirmwareStatusNotificationRequest** With **status** *Installed*	**21.** The OCTT responds with a **FirmwareStatusNotificationResponse**
	22. The Charging Station sends a **SecurityEventNotificationRequest** With **type** *FirmwareUpdated*	**23.** The OCTT responds with a **SecurityEventNotificationResponse**

Test case name	Secure Firmware Update - DownloadScheduled
Tool validations	* Step 2: Message **UpdateFirmwareResponse** - **status** *Accepted* * Step 3: Message **FirmwareStatusNotificationRequest** - **status** *DownloadScheduled* * Step 5: Message **FirmwareStatusNotificationRequest** - **status** *Downloading* * Step 7: Message **FirmwareStatusNotificationRequest** - **status** *Downloaded* * Step 9: Message **FirmwareStatusNotificationRequest** - **status** *SignatureVerified* * Step 11: Message: **StatusNotificationRequest** - **connectorStatus** *Unavailable* *Or* Message: **NotifyEventRequest** - **eventData[0].trigger** *Delta* - **eventData[0].actualValue** *"Unavailable"* - **eventData[0].component.name** *"Connector"* - **eventData[0].variable.name** *"AvailabilityState"* * Step 14: Message **FirmwareStatusNotificationRequest** - **status** *Installing* * Step 18: Message: **StatusNotificationRequest** - **connectorStatus** *Available* *Or* Message: **NotifyEventRequest** - **eventData[0].trigger** *Delta* - **eventData[0].actualValue** *"Available"* - **eventData[0].component.name** *"Connector"* - **eventData[0].variable.name** *"AvailabilityState"* * Step 20: Message **FirmwareStatusNotificationRequest** - **status** *Installed* * Step 22: Message **SecurityEventNotificationRequest** - **type** *FirmwareUpdated*
	Post scenario validations: N/a

Table 7. Test Case Id: TC_L_06_CS

Test case name	**Secure Firmware Update - InvalidSignature**	
Test case Id	TC_L_06_CS	
Use case Id(s)	L01	
Requirement(s)	L01.FR.01,L01.FR.03,L01.FR.04,L01.FR.10,L01.FR.20	
System under test	Charging Station	
Description	The CSMS is able to request the Charging Station to securely download and install a new firmware by sending an UpdateFirmwareRequest with a signingCertificate.	
Purpose	To verify if the Charging Station is able to identify if the signature is invalid and report this to the CSMS.	
Prerequisite(s)	A file server has been setup according to the (by the Charging Station) supported file transfer protocol(s), indicated by the configuration variable **FileTransferProtocols**.	
Before (Preparations)	**Configuration State:** <Configured invalid firmware signature> should be a real signature	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **UpdateFirmwareResponse**	**1.** The OCTT sends a **UpdateFirmwareRequest** with **firmware.installDateTime** *<Current DateTime - 2 hours>* **firmware.location** *<Configured firmware_location>* **firmware.retrieveDateTime** *<Current DateTime - 2 hours>* **firmware.signingCertificate** *<Configured signingCertificate>* **firmware.signature** *<Configured invalid firmware signature>*
	3. The Charging Station sends a **FirmwareStatusNotificationRequest**. With **status** *Downloading*	**4.** The OCTT responds with a **FirmwareStatusNotificationResponse**.
	5. The Charging Station sends a **FirmwareStatusNotificationRequest**. With **status** *Downloaded*	**6.** The OCTT responds with a **FirmwareStatusNotificationResponse**.
	7. The Charging Station sends a **FirmwareStatusNotificationRequest**. With **status** *InvalidSignature*	**8.** The OCTT responds with a **FirmwareStatusNotificationResponse**.
	9. The Charging Station sends a **SecurityEventNotificationRequest**. With **type** *InvalidFirmwareSignature*	**10.** The OCTT responds with a **SecurityEventNotificationResponse**.

Test case name	Secure Firmware Update - InvalidSignature
Tool validations	* Step 2: Message **UpdateFirmwareResponse** - **status** *Accepted* * Step 3: Message **FirmwareStatusNotificationRequest** - **status** *Downloading* * Step 5: Message **FirmwareStatusNotificationRequest** - **status** *Downloaded* * Step 7: Message **FirmwareStatusNotificationRequest** - **status** *InvalidSignature* * Step 9: Message **SecurityEventNotificationRequest** - **type** *InvalidFirmwareSignature*
	Post scenario validations: N/a

Table 8. Test Case Id: TC_L_07_CS

Test case name	**Secure Firmware Update - DownloadFailed**	
Test case Id	TC_L_07_CS	
Use case Id(s)	L01	
Requirement(s)	L01.FR.01,L01.FR.10,L01.FR.20	
System under test	Charging Station	
Description	The CSMS is able to request the Charging Station to securely download and install a new firmware by sending an UpdateFirmwareRequest with a signingCertificate.	
Purpose	To verify if the Charging Station is able to report to the CSMS when it is unable to download the new firmware.	
Prerequisite(s)	- A file server has been setup according to the (by the Charging Station) supported file transfer protocol(s), indicated by the configuration variable **FileTransferProtocols**. - The at the OCTT configured invalid firmware location needs to point to a not existing firmware file name.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **UpdateFirmwareResponse**	**1.** The OCTT sends a **UpdateFirmwareRequest** with **firmware.installDateTime** <*Current DateTime - 2 hours*> **firmware.location** <*Configured firmware location*> + *"_does_not_exist"* ***firmware.retrieveDateTime*** _<*Current DateTime - 2 hours*> **firmware.signingCertificate** <*Configured signingCertificate*> **firmware.signature** <*Configured signature*>
	3. The Charging Station sends a **FirmwareStatusNotificationRequest**. With **status** *Downloading* Note(s): *- This step is optional. The Charging Station may immediately identify downloading the firmware is not possible.*	**4.** The OCTT responds with a **FirmwareStatusNotificationResponse**.
	5. The Charging Station sends a **FirmwareStatusNotificationRequest**. With **status** *DownloadFailed*	**6.** The OCTT responds with a **FirmwareStatusNotificationResponse**.
Tool validations	* Step 2: Message **UpdateFirmwareResponse** - **status** *Accepted* * Step 3: Message **FirmwareStatusNotificationRequest** - **status** *Downloading* * Step 5: Message **FirmwareStatusNotificationRequest** - **status** *DownloadFailed*	
	Post scenario validations: N/a	

Table 9. Test Case Id: TC_L_08_CS

Test case name	**Secure Firmware Update - InstallVerificationFailed or InstallationFailed**	
Test case Id	TC_L_08_CS	
Use case Id(s)	L01	
Requirement(s)	L01.FR.01,L01.FR.10,L01.FR.12,L01.FR.20	
System under test	Charging Station	
Description	The CSMS is able to request the Charging Station to securely download and install a new firmware by sending an UpdateFirmwareRequest with a signingCertificate.	
Purpose	To verify if the Charging Station is able to report to the CSMS when the firmware verification fails.	
Prerequisite(s)	- A file server has been setup according to the (by the Charging Station) supported file transfer protocol(s), indicated by the configuration variable **FileTransferProtocols**. - The at the OCTT configured invalid firmware location needs to point to a firmware file that causes an InstallVerificationFailed.	
Before (Preparations)	**Configuration State:** <Configured invalid firmware location> should point to existing firmware that causes an InstallVerificationFailed <Configured invalid firmware signingCertificate> should be a trusted signingCertificate <Configured invalid firmware signature> should be a real signature	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **UpdateFirmwareResponse**	**1.** The OCTT sends a **UpdateFirmwareRequest** with **firmware.installDateTime** *<Current DateTime - 2 hours>* **firmware.location** *<Configured invalid firmware location>* **firmware.retrieveDateTime** *<Current DateTime + <Current DateTime - 2 hours>>* **firmware.signingCertificate** *<Configured invalid firmware signingCertificate>* **firmware.signature** *<Configured invalid firmware signature>*
	3. The Charging Station sends a **FirmwareStatusNotificationRequest** With **status** *Downloading*	**4.** The OCTT responds with a **FirmwareStatusNotificationResponse**
	5. The Charging Station sends a **FirmwareStatusNotificationRequest** With **status** *Downloaded*	**6.** The OCTT responds with a **FirmwareStatusNotificationResponse**
	7. The Charging Station sends a **FirmwareStatusNotificationRequest** With **status** *SignatureVerified*	**8.** The OCTT responds with a **FirmwareStatusNotificationResponse**
	9. The Charging Station notifies the CSMS about the current state of all connectors. Note(s): *- This step is optional. The Charging Station may wants to set its connectors to Unavailable, before proceeding installing the new firmware.*	**10.** The OCTT responds accordingly.
	11. Execute **Reusable State** *RebootBeforeFirmwareInstallation* Note: *This step only needs to be executed if the Charging Station needs to reboot before firmware installation.*	

Test case name	Secure Firmware Update - InstallVerificationFailed or InstallationFailed	
	12. The Charging Station sends a **FirmwareStatusNotificationRequest** With **status** *Installing* Note(s): *- This step only needs to be executed if the Charging Station did NOT reboot before firmware installation, at step 11.*	**13.** The OCTT responds with a **FirmwareStatusNotificationResponse**
	14. Execute **Reusable State** *RebootBeforeFirmwareActivation* Note: *This step only needs to be executed if the Charging Station needs to reboot before firmware activation.*	
	15. The OCTT waits for the Charging Station to reconnect. Note: *This step only needs to be executed if the Charging Station did not reboot/reconnect up until this point. The Charging Station should at least reconnect to reestablish the protocol version handshake.*	
	Note: *Step 16 through 21 can be send in a different order.*	
	16. The Charging Station notifies the CSMS about the current state of all connectors. Note(s): *- This step only needs to be executed if the connectors were previously set to Unavailable (at step 9) and the Charging Station did not report setting them back to Available (after a reboot sequence at step 11 or 14) yet.*	**17.** The OCTT responds accordingly.
	18. The Charging Station sends a **FirmwareStatusNotificationRequest** With **status** *InstallVerificationFailed* or *InstallationFailed*	**19.** The OCTT responds with a **FirmwareStatusNotificationResponse**
	20. The Charging Station sends a **SecurityEventNotificationRequest** With **type** *FirmwareUpdated*	**21.** The OCTT responds with a **SecurityEventNotificationResponse**

Test case name	Secure Firmware Update - InstallVerificationFailed or InstallationFailed
Tool validations	* Step 2: Message **UpdateFirmwareResponse** - **status** *Accepted* * Step 3: Message **FirmwareStatusNotificationRequest** - **status** *Downloading* * Step 5: Message **FirmwareStatusNotificationRequest** - **status** *Downloaded* * Step 7: Message **FirmwareStatusNotificationRequest** - **status** *SignatureVerified* * Step 9: Message: **StatusNotificationRequest** - **connectorStatus** *Unavailable* *Or* Message: **NotifyEventRequest** - **eventData[0].trigger** *Delta* - **eventData[0].actualValue** *"Unavailable"* - **eventData[0].component.name** *"Connector"* - **eventData[0].variable.name** *"AvailabilityState"* * Step 12: Message **FirmwareStatusNotificationRequest** - **status** *Installing* * Step 16: Message: **StatusNotificationRequest** - **connectorStatus** *Available* *Or* Message: **NotifyEventRequest** - **eventData[0].trigger** *Delta* - **eventData[0].actualValue** *"Available"* - **eventData[0].component.name** *"Connector"* - **eventData[0].variable.name** *"AvailabilityState"* * Step 18: Message **FirmwareStatusNotificationRequest** - **status** *InstallVerificationFailed* or *InstallationFailed* * Step 20: Message **SecurityEventNotificationRequest** - **type** *FirmwareUpdated*
	Post scenario validations: N/a

Table 10. Test Case Id: TC_L_10_CS

Test case name	**Secure Firmware Update - AcceptedCanceled**
Test case Id	TC_L_10_CS
Use case Id(s)	L01
Requirement(s)	L01.FR.01,L01.FR.10,L01.FR.20,L01.FR.24
System under test	Charging Station
Description	The CSMS is able to request the Charging Station to securely download and install a new firmware by sending an UpdateFirmwareRequest with a signingCertificate.
Purpose	To verify if the Charging Station is able to cancel an ongoing firmware update and start a new one, when receiving an UpdateFirmwareRequest from the CSMS.
Prerequisite(s)	- A file server has been setup according to the (by the Charging Station) supported file transfer protocol(s), indicated by the configuration variable **FileTransferProtocols**. - The Charging Station is able to cancel an ongoing firmware update while it is busy downloading a new firmware file.
Before (Preparations)	**Configuration State:** N/a
	Memory State: N/a
	Reusable State(s): N/a

Test case name	Secure Firmware Update - AcceptedCanceled	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **UpdateFirmwareResponse** With **status** *Accepted*	**1.** The OCTT sends a **UpdateFirmwareRequest** with **firmware.installDateTime** *<Current DateTime - 2 hours>* **firmware.location** *<Configured firmware_location>* **firmware.retrieveDateTime** *<Current DateTime - 2 hours>* **firmware.signingCertificate** *<Configured signingCertificate>* **firmware.signature** *<Configured signature>*
	3. The Charging Station sends a **FirmwareStatusNotificationRequest** With **status** *Downloading*	**4.** The OCTT responds with a **FirmwareStatusNotificationResponse**
	6. The Charging Station responds with a **UpdateFirmwareResponse** With **status** *AcceptedCanceled*	**5.** The OCTT sends a **UpdateFirmwareRequest** with **firmware.installDateTime** *<Current DateTime - 2 hours>* **firmware.location** *<Configured firmware_location>* **firmware.retrieveDateTime** *<Current DateTime - 2 hours>* **firmware.signingCertificate** *<Configured signingCertificate>* **firmware.signature** *<Configured signature>*
	7. The Charging Station sends a **FirmwareStatusNotificationRequest** With **status** *Downloading*	**8.** The OCTT responds with a **FirmwareStatusNotificationResponse**
	9. The Charging Station sends a **FirmwareStatusNotificationRequest** With **status** *Downloaded*	**10.** The OCTT responds with a **FirmwareStatusNotificationResponse**
	11. The Charging Station sends a **FirmwareStatusNotificationRequest** With **status** *SignatureVerified*	**12.** The OCTT responds with a **FirmwareStatusNotificationResponse**
	13. The Charging Station notifies the CSMS about the current state of all connectors. Note(s): *- This step is optional. The Charging Station may wants to set its connectors to Unavailable, before proceeding installing the new firmware.*	**14.** The OCTT responds accordingly.
	15. Execute **Reusable State** *RebootBeforeFirmwareInstallation* Note: *This step only needs to be executed if the Charging Station needs to reboot before firmware installation.*	
	16. The Charging Station sends a **FirmwareStatusNotificationRequest** With **status** *Installing* Note(s): *- This step only needs to be executed if the Charging Station did NOT reboot before firmware installation, at step 15.*	**17.** The OCTT responds with a **FirmwareStatusNotificationResponse**
	18. Execute **Reusable State** *RebootBeforeFirmwareActivation* Note: *This step only needs to be executed if the Charging Station needs to reboot before firmware activation.*	

Test case name	Secure Firmware Update - AcceptedCanceled	
	19. The OCTT waits for the Charging Station to reconnect. Note: *This step only needs to be executed if the Charging Station did not reboot/reconnect up until this point. The Charging Station should at least reconnect to reestablish the protocol version handshake.*	
	Note: *Step 20 through 25 can be send in a different order.*	
	20. The Charging Station notifies the CSMS about the current state of all connectors. Note(s): *- This step only needs to be executed if the connectors were previously set to Unavailable (at step 13) and the Charging Station did not report setting them back to Available (after a reboot sequence at step 15 or 18) yet.*	**21.** The OCTT responds accordingly.
	22. The Charging Station sends a **FirmwareStatusNotificationRequest** With **status** *Installed*	**23.** The OCTT responds with a **FirmwareStatusNotificationResponse**
	24. The Charging Station sends a **SecurityEventNotificationRequest** With **type** *FirmwareUpdated*	**25.** The OCTT responds with a **SecurityEventNotificationResponse**

Test case name	Secure Firmware Update - AcceptedCanceled
Tool validations	* Step 2: Message **UpdateFirmwareResponse** - **status** *Accepted* * Step 3: Message **FirmwareStatusNotificationRequest** - **status** *Downloading* * Step 6: Message **UpdateFirmwareResponse** - **status** *AcceptedCanceled* (The requestId at the **FirmwareStatusNotificationRequest** messages must refer to the one from the second **UpdateFirmwareRequest** from this point on). * Step 7: Message FirmwareStatusNotificationRequest - **status** *Downloading* * Step 9: Message **FirmwareStatusNotificationRequest** - **status** *Downloaded* * Step 11: Message **FirmwareStatusNotificationRequest** - **status** *SignatureVerified* * Step 13: Message: **StatusNotificationRequest** - **connectorStatus** *Unavailable* *Or* Message: **NotifyEventRequest** - **eventData[0].trigger** *Delta* - **eventData[0].actualValue** *"Unavailable"* - **eventData[0].component.name** *"Connector"* - **eventData[0].variable.name** *"AvailabilityState"* * Step 16: Message **FirmwareStatusNotificationRequest** - **status** *Installing* * Step 20: Message: **StatusNotificationRequest** - **connectorStatus** *Available* *Or* Message: **NotifyEventRequest** - **eventData[0].trigger** *Delta* - **eventData[0].actualValue** *"Available"* - **eventData[0].component.name** *"Connector"* - **eventData[0].variable.name** *"AvailabilityState"* * Step 22: Message **FirmwareStatusNotificationRequest** - **status** *Installed* * Step 24: Message **SecurityEventNotificationRequest** - **type** *FirmwareUpdated*
	Post scenario validations: N/a

Table 11. Test Case Id: TC_L_11_CS

Test case name	**Secure Firmware Update - Unable to cancel**	
Test case Id	TC_L_11_CS	
Use case Id(s)	L01	
Requirement(s)	L01.FR.01,L01.FR.10,L01.FR.20,L01.FR.27	
System under test	Charging Station	
Description	The CSMS is able to request the Charging Station to securely download and install a new firmware by sending an UpdateFirmwareRequest with a signingCertificate.	
Purpose	To verify if the Charging Station is able to reject a firmware update request when it is unable to cancel an ongoing firmware update.	
Prerequisite(s)	- A file server has been setup according to the (by the Charging Station) supported file transfer protocol(s), indicated by the configuration variable **FileTransferProtocols**. - The Charging Station is NOT able to cancel an ongoing firmware update.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **UpdateFirmwareResponse** With **status** *Accepted*	**1.** The OCTT sends a **UpdateFirmwareRequest** with **firmware.installDateTime** *<Current DateTime - 2 hours>* **firmware.location** *<Configured firmware_location>* **firmware.retrieveDateTime** *<Current DateTime - 2 hours>* **firmware.signingCertificate** *<Configured signingCertificate>* **firmware.signature** *<Configured signature>*
	3. The Charging Station sends a **FirmwareStatusNotificationRequest** With **status** *Downloading*	**4.** The OCTT responds with a **FirmwareStatusNotificationResponse**
	6. The Charging Station responds with a **UpdateFirmwareResponse** With **status** *Rejected*	**5.** The OCTT sends a **UpdateFirmwareRequest** with **firmware.installDateTime** *<Current DateTime - 2 hours>* **firmware.location** *<Configured firmware_location>* **firmware.retrieveDateTime** *<Current DateTime - 2 hours>* **firmware.signingCertificate** *<Configured signingCertificate>* **firmware.signature** *<Configured signature>*
	7. The Charging Station sends a **FirmwareStatusNotificationRequest** With **status** *Downloaded*	**8.** The OCTT responds with a **FirmwareStatusNotificationResponse**
	9. The Charging Station sends a **FirmwareStatusNotificationRequest** With **status** *SignatureVerified*	**10.** The OCTT responds with a **FirmwareStatusNotificationResponse**
	11. The Charging Station notifies the CSMS about the current state of all connectors. Note(s): *- This step is optional. The Charging Station may wants to set its connectors to Unavailable, before proceeding installing the new firmware.*	**12.** The OCTT responds accordingly.

Test case name	Secure Firmware Update - Unable to cancel	
	13. Execute **Reusable State** *RebootBeforeFirmwareInstallation* Note: *This step only needs to be executed if the Charging Station needs to reboot before firmware installation.*	
	14. The Charging Station sends a **FirmwareStatusNotificationRequest** With **status** *Installing* Note(s): *- This step only needs to be executed if the Charging Station did NOT reboot before firmware installation, at step 13.*	**15.** The OCTT responds with a **FirmwareStatusNotificationResponse**
	16. Execute **Reusable State** *RebootBeforeFirmwareActivation* Note: *This step only needs to be executed if the Charging Station needs to reboot before firmware activation.*	
	17. The OCTT waits for the Charging Station to reconnect. Note: *This step only needs to be executed if the Charging Station did not reboot/reconnect up until this point. The Charging Station should at least reconnect to reestablish the protocol version handshake.*	
	Note: *Step 18 through 23 can be send in a different order.*	
	18. The Charging Station notifies the CSMS about the current state of all connectors. Note(s): *- This step only needs to be executed if the connectors were previously set to Unavailable (at step 11) and the Charging Station did not report setting them back to Available (after a reboot sequence at step 13 or 16) yet.*	**19.** The OCTT responds accordingly.
	20. The Charging Station sends a **FirmwareStatusNotificationRequest** With **status** *Installed*	**21.** The OCTT responds with a **FirmwareStatusNotificationResponse**
	22. The Charging Station sends a **SecurityEventNotificationRequest** With **type** *FirmwareUpdated*	**23.** The OCTT responds with a **SecurityEventNotificationResponse**

Test case name	Secure Firmware Update - Unable to cancel
Tool validations	* Step 2: Message **UpdateFirmwareResponse** - **status** *Accepted* * Step 3: Message **FirmwareStatusNotificationRequest** - **status** *Downloading* * Step 6: Message **UpdateFirmwareResponse** - **status** *Rejected* * Step 7: Message **FirmwareStatusNotificationRequest** - **status** *Downloaded* * Step 9: Message **FirmwareStatusNotificationRequest** - **status** *SignatureVerified* * Step 11: Message: **StatusNotificationRequest** - **connectorStatus** *Unavailable* *Or* Message: **NotifyEventRequest** - **eventData[0].trigger** *Delta* - **eventData[0].actualValue** *"Unavailable"* - **eventData[0].component.name** *"Connector"* - **eventData[0].variable.name** *"AvailabilityState"* * Step 14: Message **FirmwareStatusNotificationRequest** - **status** *Installing* * Step 18: Message: **StatusNotificationRequest** - **connectorStatus** *Available* *Or* Message: **NotifyEventRequest** - **eventData[0].trigger** *Delta* - **eventData[0].actualValue** *"Available"* - **eventData[0].component.name** *"Connector"* - **eventData[0].variable.name** *"AvailabilityState"* * Step 20: Message **FirmwareStatusNotificationRequest** - **status** *Installed* * Step 22: Message **SecurityEventNotificationRequest** - **type** *FirmwareUpdated*
	Post scenario validations: N/a

Table 12. Test Case Id: TC_L_12_CS

Test case name	**Secure Firmware Update - Unable to download/install firmware with ongoing transaction - AllowNewSessionsPendingFirmwareUpdate is true**	
Test case Id	TC_L_12_CS	
Use case Id(s)	L01	
Requirement(s)	L01.FR.01,L01.FR.06,L01.FR.07,L01.FR.10,L01.FR.20	
System under test	Charging Station	
Description	The CSMS is able to request the Charging Station to securely download and install a new firmware by sending an UpdateFirmwareRequest with a signingCertificate.	
Purpose	To verify if the Charging Station is able to keep allowing new transactions when requested to update the firmware, while there is an ongoing transaction.	
Prerequisite(s)	- A file server has been setup according to the (by the Charging Station) supported file transfer protocol(s), indicated by the configuration variable **FileTransferProtocols**. - The Charging Station is able to start more than one transaction at a time. - The Charging Station is unable to download AND install firmware while there is an ongoing transaction.	
Before (Preparations)	**Configuration State:** **AllowNewSessionsPendingFirmwareUpdate** is *true* (If implemented)	
	Memory State: N/a	
	Reusable State(s): **State is** *EnergyTransferStarted* for *<Configured connectorId>*	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **UpdateFirmwareResponse** With **status** *Accepted*	**1.** The OCTT sends a **UpdateFirmwareRequest** with **firmware.installDateTime** *<Current DateTime - 2 hours>* **firmware.location** *<Configured firmware_location>* **firmware.retrieveDateTime** *<Current DateTime - 2 hours>* **firmware.signingCertificate** *<Configured signingCertificate>* **firmware.signature** *<Configured signature>*
	3. The Charging Station sends a **FirmwareStatusNotificationRequest** With **status** *DownloadScheduled*	**4.** The OCTT responds with a **FirmwareStatusNotificationResponse**
	5. Execute **Reusable State** *EnergyTransferStarted* for *<Configured second Connector>* Note(s): *- It is allowed to start a second transaction while there is a scheduled firmware update.*	
	6. Execute **Reusable State** *ParkingBayUnoccupied* for *<Configured connectorId>* Note(s): *- The Charging Station will proceed to this end state. This will cause the transaction to stop.*	
	7. Execute **Reusable State** *ParkingBayUnoccupied* for *<Configured second Connector>* Note(s): *- The Charging Station will proceed to this end state. This will cause the transaction to stop.* *- The Charging Station will start the firmware update process the moment this second transaction ends or when all interactions with the EV Driver are done (So after the cable has been unplugged, if there is no parking bay sensor).*	

Test case name	Secure Firmware Update - Unable to download/install firmware with ongoing transaction - AllowNewSessionsPendingFirmwareUpdate is true	
	8. The Charging Station sends a **FirmwareStatusNotificationRequest** With **status** *Downloading*	**9.** The OCTT responds with a **FirmwareStatusNotificationResponse**
	10. The Charging Station sends a **FirmwareStatusNotificationRequest** With **status** *Downloaded*	**11.** The OCTT responds with a **FirmwareStatusNotificationResponse**
	12. The Charging Station sends a **FirmwareStatusNotificationRequest** With **status** *SignatureVerified*	**13.** The OCTT responds with a **FirmwareStatusNotificationResponse**
	14. The Charging Station notifies the CSMS about the current state of all connectors. Note(s): *- This step is optional. The Charging Station may wants to set its connectors to Unavailable, before proceeding installing the new firmware.*	**15.** The OCTT responds accordingly.
	16. Execute **Reusable State** *RebootBeforeFirmwareInstallation* Note: *This step only needs to be executed if the Charging Station needs to reboot before firmware installation.*	
	17. The Charging Station sends a **FirmwareStatusNotificationRequest** With **status** *Installing* Note(s): *- This step only needs to be executed if the Charging Station did NOT reboot before firmware installation, at step 16.*	**18.** The OCTT responds with a **FirmwareStatusNotificationResponse**
	19. Execute **Reusable State** *RebootBeforeFirmwareActivation* Note: *This step only needs to be executed if the Charging Station needs to reboot before firmware activation.*	

Test case name	Secure Firmware Update - Unable to download/install firmware with ongoing transaction - AllowNewSessionsPendingFirmwareUpdate is true	
	20. The OCTT waits for the Charging Station to reconnect. Note: *This step only needs to be executed if the Charging Station did not reboot/reconnect up until this point. The Charging Station should at least reconnect to reestablish the protocol version handshake.*	
	Note: *Step 21 through 26 can be send in a different order.*	
	21. The Charging Station notifies the CSMS about the current state of all connectors. Note(s): *- This step only needs to be executed if the connectors were previously set to Unavailable (at step 14) and the Charging Station did not report setting them back to Available (after a reboot sequence at step 16 or 19) yet.*	**22.** The OCTT responds accordingly.
	23. The Charging Station sends a **FirmwareStatusNotificationRequest** With **status** *Installed*	**24.** The OCTT responds with a **FirmwareStatusNotificationResponse**
	25. The Charging Station sends a **SecurityEventNotificationRequest** With **type** *FirmwareUpdated*	**26.** The OCTT responds with a **SecurityEventNotificationResponse**

Test case name	Secure Firmware Update - Unable to download/install firmware with ongoing transaction - AllowNewSessionsPendingFirmwareUpdate is true
Tool validations	* Step 2: Message **UpdateFirmwareResponse** - **status** *Accepted* * Step 3: Message **FirmwareStatusNotificationRequest** - **status** *DownloadScheduled* * Step 8: Message **FirmwareStatusNotificationRequest** - **status** *Downloading* * Step 10: Message **FirmwareStatusNotificationRequest** - **status** *Downloaded* * Step 12: Message **FirmwareStatusNotificationRequest** - **status** *SignatureVerified* * Step 14: Message: **StatusNotificationRequest** - **connectorStatus** *Unavailable* *Or* Message: **NotifyEventRequest** - **eventData[0].trigger** *Delta* - **eventData[0].actualValue** *"Unavailable"* - **eventData[0].component.name** *"Connector"* - **eventData[0].variable.name** *"AvailabilityState"* * Step 17: Message **FirmwareStatusNotificationRequest** - **status** *Installing* * Step 21: Message: **StatusNotificationRequest** - **connectorStatus** *Available* *Or* Message: **NotifyEventRequest** - **eventData[0].trigger** *Delta* - **eventData[0].actualValue** *"Available"* - **eventData[0].component.name** *"Connector"* - **eventData[0].variable.name** *"AvailabilityState"* * Step 23: Message **FirmwareStatusNotificationRequest** - **status** *Installed* * Step 25: Message **SecurityEventNotificationRequest** - **type** *FirmwareUpdated*
	Post scenario validations: N/a

Table 13. Test Case Id: TC_L_13_CS

Test case name	**Secure Firmware Update - Unable to download/install firmware with ongoing transaction - AllowNewSessionsPendingFirmwareUpdate is false**	
Test case Id	TC_L_13_CS	
Use case Id(s)	L01	
Requirement(s)	L01.FR.01,L01.FR.06,L01.FR.07,L01.FR.10,L01.FR.20	
System under test	Charging Station	
Description	The CSMS is able to request the Charging Station to securely download and install a new firmware by sending an UpdateFirmwareRequest with a signingCertificate.	
Purpose	To verify if the Charging Station is able to set its available connectors to Unavailable when requested to update the firmware, while there is an ongoing transaction.	
Prerequisite(s)	- A file server has been setup according to the (by the Charging Station) supported file transfer protocol(s), indicated by the configuration variable **FileTransferProtocols**. - The configuration variable **AllowNewSessionsPendingFirmwareUpdate** is implemented. - The Charging Station is unable to download AND install firmware while there is an ongoing transaction.	
Before (Preparations)	**Configuration State:** **AllowNewSessionsPendingFirmwareUpdate** is *false*	
	Memory State: N/a	
	Reusable State(s): **State is** *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **UpdateFirmwareResponse** With **status** *Accepted*	**1.** The OCTT sends a **UpdateFirmwareRequest** with **firmware.installDateTime** *<Current DateTime - 2 hours>* **firmware.location** *<Configured firmware_location>* **firmware.retrieveDateTime** *<Current DateTime - 2 hours>* **firmware.signingCertificate** *<Configured signingCertificate>* **firmware.signature** *<Configured signature>*
	3. The Charging Station sends a **FirmwareStatusNotificationRequest** With **status** *DownloadScheduled* Note: *This step is optional. Part 2 specification only describes that this status needs to be send in case the retrieveDateTime is in the future. However it is also allowed to send this status if the Charging Station schedules the firmware download, because of an ongoing transaction.*	**4.** The OCTT responds with a **FirmwareStatusNotificationResponse**
	5. The Charging Station notifies the CSMS about the current state of its Available connector(s). Note(s): *- This step needs to be executed for all connectors with AvailabilityState Available.*	**6.** The OCTT responds accordingly.
	7. Execute **Reusable State** *ParkingBayUnoccupied* for *<Configured connectorId>* Note(s): *- The Charging Station will proceed to this end state. This will cause the transaction to stop.* *- The Charging Station will start the firmware update process the moment the transaction ends or when all interactions with the EV Driver are done (So after the cable has been unplugged, if there is no parking bay sensor).*	

Test case name	Secure Firmware Update - Unable to download/install firmware with ongoing transaction - AllowNewSessionsPendingFirmwareUpdate is false	
	8. The Charging Station sends a **FirmwareStatusNotificationRequest** With **status** *Downloading*	**9.** The OCTT responds with a **FirmwareStatusNotificationResponse**
	10. The Charging Station sends a **FirmwareStatusNotificationRequest** With **status** *Downloaded*	**11.** The OCTT responds with a **FirmwareStatusNotificationResponse**
	12. The Charging Station sends a **FirmwareStatusNotificationRequest** With **status** *SignatureVerified*	**13.** The OCTT responds with a **FirmwareStatusNotificationResponse**
	14. The Charging Station notifies the CSMS about the current state of all connectors. Note(s): *- This step is optional. The Charging Station may wants to set its last connector also to Unavailable, before proceeding installing the new firmware.*	**15.** The OCTT responds accordingly.
	16. Execute **Reusable State** *RebootBeforeFirmwareInstallation* Note: *This step only needs to be executed if the Charging Station needs to reboot before firmware installation.*	
	17. The Charging Station sends a **FirmwareStatusNotificationRequest** With **status** *Installing* Note(s): *- This step only needs to be executed if the Charging Station did NOT reboot before firmware installation, at step 16.*	**18.** The OCTT responds with a **FirmwareStatusNotificationResponse**
	19. Execute **Reusable State** *RebootBeforeFirmwareActivation* Note: *This step only needs to be executed if the Charging Station needs to reboot before firmware activation.*	

Test case name	Secure Firmware Update - Unable to download/install firmware with ongoing transaction - AllowNewSessionsPendingFirmwareUpdate is false	
	20. The OCTT waits for the Charging Station to reconnect. Note: *This step only needs to be executed if the Charging Station did not reboot/reconnect up until this point. The Charging Station should at least reconnect to reestablish the protocol version handshake.*	
	Note: *Step 21 through 26 can be send in a different order.*	
	21. The Charging Station notifies the CSMS about the current state of all connectors. Note(s): *- This step only needs to be executed if the connectors were previously set to Unavailable (at step 14) and the Charging Station did not report setting them back to Available (after a reboot sequence at step 16 or 19) yet.*	**22.** The OCTT responds accordingly.
	23. The Charging Station sends a **FirmwareStatusNotificationRequest** With **status** *Installed*	**24.** The OCTT responds with a **FirmwareStatusNotificationResponse**
	25. The Charging Station sends a **SecurityEventNotificationRequest** With **type** *FirmwareUpdated*	**26.** The OCTT responds with a **SecurityEventNotificationResponse**

Test case name	Secure Firmware Update - Unable to download/install firmware with ongoing transaction - AllowNewSessionsPendingFirmwareUpdate is false
Tool validations	* Step 2: Message **UpdateFirmwareResponse** - **status** *Accepted* * Step 3: Message **FirmwareStatusNotificationRequest** - **status** *DownloadScheduled* * Step 5: Message: **StatusNotificationRequest** - **connectorStatus** *Unavailable* *Or* Message: **NotifyEventRequest** - **eventData[0].trigger** *Delta* - **eventData[0].actualValue** *"Unavailable"* - **eventData[0].component.name** *"Connector"* - **eventData[0].variable.name** *"AvailabilityState"* * Step 8: Message **FirmwareStatusNotificationRequest** - **status** *Downloading* * Step 10: Message **FirmwareStatusNotificationRequest** - **status** *Downloaded* * Step 12: Message **FirmwareStatusNotificationRequest** - **status** *SignatureVerified* * Step 14: Message: **StatusNotificationRequest** - **connectorStatus** *Unavailable* *Or* Message: **NotifyEventRequest** - **eventData[0].trigger** *Delta* - **eventData[0].actualValue** *"Unavailable"* - **eventData[0].component.name** *"Connector"* - **eventData[0].variable.name** *"AvailabilityState"* * Step 17: Message **FirmwareStatusNotificationRequest** - **status** *Installing* * Step 21: Message: **StatusNotificationRequest** - **connectorStatus** *Available* *Or* Message: **NotifyEventRequest** - **eventData[0].trigger** *Delta* - **eventData[0].actualValue** *"Available"* - **eventData[0].component.name** *"Connector"* - **eventData[0].variable.name** *"AvailabilityState"* * Step 23: Message **FirmwareStatusNotificationRequest** - **status** *Installed* * Step 25: Message **SecurityEventNotificationRequest** - **type** *FirmwareUpdated*
	Post scenario validations: N/a

Table 14. Test Case Id: TC_L_14_CS

Test case name	**Secure Firmware Update - Unable to install firmware with ongoing transaction - AllowNewSessionsPendingFirmwareUpdate is true**
Test case Id	TC_L_14_CS
Use case Id(s)	L01
Requirement(s)	L01.FR.01,L01.FR.06,L01.FR.07,L01.FR.10,L01.FR.20
System under test	Charging Station
Description	The CSMS is able to request the Charging Station to securely download and install a new firmware by sending an UpdateFirmwareRequest with a signingCertificate.
Purpose	To verify if the Charging Station is able to keep allowing new transactions when requested to update the firmware, while there is an ongoing transaction.
Prerequisite(s)	- A file server has been setup according to the (by the Charging Station) supported file transfer protocol(s), indicated by the configuration variable **FileTransferProtocols**. - The Charging Station is able to start more than one transaction at a time. - The Charging Station is unable to install firmware while there is an ongoing transaction.
Before (Preparations)	**Configuration State:** **AllowNewSessionsPendingFirmwareUpdate** is *true* (If implemented)
	Memory State: N/a
	Reusable State(s): **State is** *EnergyTransferStarted* for EVSEId *1* and ConnectorId *1*

Test case name	Secure Firmware Update - Unable to install firmware with ongoing transaction - AllowNewSessionsPendingFirmwareUpdate is true	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **UpdateFirmwareResponse** With **status** *Accepted*	**1.** The OCTT sends a **UpdateFirmwareRequest** with **firmware.installDateTime** *<Current DateTime - 2 hours>* **firmware.location** *<Configured firmware_location>* **firmware.retrieveDateTime** *<Current DateTime - 2 hours>* **firmware.signingCertificate** *<Configured signingCertificate>* **firmware.signature** *<Configured signature>*
	3. The Charging Station sends a **FirmwareStatusNotificationRequest** With **status** *Downloading*	**4.** The OCTT responds with a **FirmwareStatusNotificationResponse**
	5. The Charging Station sends a **FirmwareStatusNotificationRequest** With **status** *Downloaded*	**6.** The OCTT responds with a **FirmwareStatusNotificationResponse**
	7. The Charging Station sends a **FirmwareStatusNotificationRequest** With **status** *SignatureVerified*	**8.** The OCTT responds with a **FirmwareStatusNotificationResponse**
	9. The Charging Station sends a **FirmwareStatusNotificationRequest** With **status** *InstallScheduled*	**10.** The OCTT responds with a **FirmwareStatusNotificationResponse**
	11. Execute **Reusable State** *EnergyTransferStarted* for *<Configured second Connector>* Note(s): *- It is allowed to start a second transaction while there is a scheduled firmware update.*	
	12. Execute **Reusable State** *ParkingBayUnoccupied* for *<Configured connectorId>* Note(s): *- The Charging Station will proceed to this end state. This will cause the transaction to stop.*	
	13. Execute **Reusable State** *ParkingBayUnoccupied* for *<Configured second Connector>* Note(s): *- The Charging Station will proceed to this end state. This will cause the transaction to stop.* *- The Charging Station will start the firmware update process the moment this second transaction ends or when all interactions with the EV Driver are done (So after the cable has been unplugged, if there is no parking bay sensor).*	
	14. The Charging Station notifies the CSMS about the current state of all connectors. Note(s): *- This step is optional. The Charging Station may wants to set its connectors to Unavailable, before proceeding installing the new firmware.*	**15.** The OCTT responds accordingly.
	16. Execute **Reusable State** *RebootBeforeFirmwareInstallation* Note: *This step only needs to be executed if the Charging Station needs to reboot before firmware installation.*	

Test case name	Secure Firmware Update - Unable to install firmware with ongoing transaction - AllowNewSessionsPendingFirmwareUpdate is true	
	17. The Charging Station sends a **FirmwareStatusNotificationRequest** With **status** *Installing* Note(s): *- This step only needs to be executed if the Charging Station did NOT reboot before firmware installation, at step 16.*	**18.** The OCTT responds with a **FirmwareStatusNotificationResponse**
	19. Execute **Reusable State** *RebootBeforeFirmwareActivation* Note: *This step only needs to be executed if the Charging Station needs to reboot before firmware activation.*	
	20. The OCTT waits for the Charging Station to reconnect. Note: *This step only needs to be executed if the Charging Station did not reboot/reconnect up until this point. The Charging Station should at least reconnect to reestablish the protocol version handshake.*	
	Note: *Step 21 through 26 can be send in a different order.*	
	21. The Charging Station notifies the CSMS about the current state of all connectors. Note(s): *- This step only needs to be executed if the connectors were previously set to Unavailable (at step 14) and the Charging Station did not report setting them back to Available (after a reboot sequence at step 16 or 19) yet.*	**22.** The OCTT responds accordingly.
	23. The Charging Station sends a **FirmwareStatusNotificationRequest** With **status** *Installed*	**24.** The OCTT responds with a **FirmwareStatusNotificationResponse**
	25. The Charging Station sends a **SecurityEventNotificationRequest** With **type** *FirmwareUpdated*	**26.** The OCTT responds with a **SecurityEventNotificationResponse**

Test case name	Secure Firmware Update - Unable to install firmware with ongoing transaction - AllowNewSessionsPendingFirmwareUpdate is true
Tool validations	* Step 2: Message **UpdateFirmwareResponse** - **status** *Accepted* * Step 3: Message **FirmwareStatusNotificationRequest** - **status** *Downloading* * Step 5: Message **FirmwareStatusNotificationRequest** - **status** *Downloaded* * Step 7: Message **FirmwareStatusNotificationRequest** - **status** *SignatureVerified* * Step 9: Message **FirmwareStatusNotificationRequest** - **status** *InstallScheduled* * Step 14: Message: **StatusNotificationRequest** - **connectorStatus** *Unavailable* *Or* Message: **NotifyEventRequest** - **eventData[0].trigger** *Delta* - **eventData[0].actualValue** *"Unavailable"* - **eventData[0].component.name** *"Connector"* - **eventData[0].variable.name** *"AvailabilityState"* * Step 17: Message **FirmwareStatusNotificationRequest** - **status** *Installing* * Step 21: Message: **StatusNotificationRequest** - **connectorStatus** *Available* *Or* Message: **NotifyEventRequest** - **eventData[0].trigger** *Delta* - **eventData[0].actualValue** *"Available"* - **eventData[0].component.name** *"Connector"* - **eventData[0].variable.name** *"AvailabilityState"* * Step 23: Message **FirmwareStatusNotificationRequest** - **status** *Installed* * Step 25: Message **SecurityEventNotificationRequest** - **type** *FirmwareUpdated*
	Post scenario validations: N/a

Table 15. Test Case Id: TC_L_15_CS

Test case name	**Secure Firmware Update - Unable to install firmware with ongoing transaction - AllowNewSessionsPendingFirmwareUpdate is false**	
Test case Id	TC_L_15_CS	
Use case Id(s)	L01	
Requirement(s)	L01.FR.01,L01.FR.06,L01.FR.07,L01.FR.10,L01.FR.20	
System under test	Charging Station	
Description	The CSMS is able to request the Charging Station to securely download and install a new firmware by sending an UpdateFirmwareRequest with a signingCertificate.	
Purpose	To verify if the Charging Station is able to set its available connectors to Unavailable when requested to update the firmware, while there is an ongoing transaction.	
Prerequisite(s)	- A file server has been setup according to the (by the Charging Station) supported file transfer protocol(s), indicated by the configuration variable **FileTransferProtocols**. - The configuration variable **AllowNewSessionsPendingFirmwareUpdate** is implemented. - The Charging Station is unable to install firmware while there is an ongoing transaction.	
Before (Preparations)	**Configuration State:** **AllowNewSessionsPendingFirmwareUpdate** is *false*	
	Memory State: N/a	
	Reusable State(s): **State is** *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **UpdateFirmwareResponse** With **status** *Accepted*	**1.** The OCTT sends a **UpdateFirmwareRequest** with **firmware.installDateTime** *<Current DateTime - 2 hours>* **firmware.location** *<Configured firmware_location>* **firmware.retrieveDateTime** *<Current DateTime - 2 hours>* **firmware.signingCertificate** *<Configured signingCertificate>* **firmware.signature** *<Configured signature>*
	3. The Charging Station sends a **FirmwareStatusNotificationRequest** With **status** *Downloading*	**4.** The OCTT responds with a **FirmwareStatusNotificationResponse**
	5. The Charging Station sends a **FirmwareStatusNotificationRequest** With **status** *Downloaded*	**6.** The OCTT responds with a **FirmwareStatusNotificationResponse**
	7. The Charging Station sends a **FirmwareStatusNotificationRequest** With **status** *SignatureVerified*	**8.** The OCTT responds with a **FirmwareStatusNotificationResponse**
	9. The Charging Station sends a **FirmwareStatusNotificationRequest** With **status** *InstallScheduled* Note: *This step is optional. Part 2 specification only describes that this status needs to be send in case the installDateTime is in the future. However it is also allowed to send this status if the Charging Station schedules the firmware installation, because of an ongoing transaction.*	**10.** The OCTT responds with a **FirmwareStatusNotificationResponse**
	11. The Charging Station notifies the CSMS about the current state of its Available connector(s). Note(s): *- This step needs to be executed for all connectors with AvailabilityState Available.*	**12.** The OCTT responds accordingly.

Test case name	Secure Firmware Update - Unable to install firmware with ongoing transaction - AllowNewSessionsPendingFirmwareUpdate is false	
	13. Execute **Reusable State** *ParkingBayUnoccupied* for *<Configured connectorId>* Note(s): *- The Charging Station will proceed to this end state. This will cause the transaction to stop.* *- The Charging Station will start the firmware update process the moment the transaction ends or when all interactions with the EV Driver are done (So after the cable has been unplugged, if there is no parking bay sensor).*	
	14. The Charging Station notifies the CSMS about the current state of all connectors. Note(s): *- This step is optional. The Charging Station may wants to set its last connector to Unavailable, before proceeding installing the new firmware.*	**15.** The OCTT responds accordingly.
	16. Execute **Reusable State** *RebootBeforeFirmwareInstallation* Note: *This step only needs to be executed if the Charging Station needs to reboot before firmware installation.*	
	17. The Charging Station sends a **FirmwareStatusNotificationRequest** With **status** *Installing* Note(s): *- This step only needs to be executed if the Charging Station did NOT reboot before firmware installation, at step 16.*	**18.** The OCTT responds with a **FirmwareStatusNotificationResponse**
	19. Execute **Reusable State** *RebootBeforeFirmwareActivation* Note: *This step only needs to be executed if the Charging Station needs to reboot before firmware activation.*	
	20. The OCTT waits for the Charging Station to reconnect. Note: *This step only needs to be executed if the Charging Station did not reboot/reconnect up until this point. The Charging Station should at least reconnect to reestablish the protocol version handshake.*	
	Note: *Step 21 through 26 can be send in a different order.*	
	21. The Charging Station notifies the CSMS about the current state of all connectors. Note(s): *- This step only needs to be executed if the connectors were previously set to Unavailable (at step 14) and the Charging Station did not report setting them back to Available (after a reboot sequence at step 16 or 19) yet.*	**22.** The OCTT responds accordingly.
	23. The Charging Station sends a **FirmwareStatusNotificationRequest** With **status** *Installed*	**24.** The OCTT responds with a **FirmwareStatusNotificationResponse**
	25. The Charging Station sends a **SecurityEventNotificationRequest** With **type** *FirmwareUpdated*	**26.** The OCTT responds with a **SecurityEventNotificationResponse**

Test case name	Secure Firmware Update - Unable to install firmware with ongoing transaction - AllowNewSessionsPendingFirmwareUpdate is false
Tool validations	* Step 2: Message **UpdateFirmwareResponse** - **status** *Accepted* * Step 3: Message **FirmwareStatusNotificationRequest** - **status** *Downloading* * Step 5: Message **FirmwareStatusNotificationRequest** - **status** *Downloaded* * Step 7: Message **FirmwareStatusNotificationRequest** - **status** *SignatureVerified* * Step 9: Message **FirmwareStatusNotificationRequest** - **status** *InstallScheduled* * Step 11: Message: **StatusNotificationRequest** - **connectorStatus** *Unavailable* *Or* Message: **NotifyEventRequest** - **eventData[0].trigger** *Delta* - **eventData[0].actualValue** *"Unavailable"* - **eventData[0].component.name** *"Connector"* - **eventData[0].variable.name** *"AvailabilityState"* * Step 14: Message: **StatusNotificationRequest** - **connectorStatus** *Unavailable* *Or* Message: **NotifyEventRequest** - **eventData[0].trigger** *Delta* - **eventData[0].actualValue** *"Unavailable"* - **eventData[0].component.name** *"Connector"* - **eventData[0].variable.name** *"AvailabilityState"* * Step 17: Message **FirmwareStatusNotificationRequest** - **status** *Installing* * Step 21: Message: **StatusNotificationRequest** - **connectorStatus** *Available* *Or* Message: **NotifyEventRequest** - **eventData[0].trigger** *Delta* - **eventData[0].actualValue** *"Available"* - **eventData[0].component.name** *"Connector"* - **eventData[0].variable.name** *"AvailabilityState"* * Step 23: Message **FirmwareStatusNotificationRequest** - **status** *Installed* * Step 25: Message **SecurityEventNotificationRequest** - **type** *FirmwareUpdated*
	Post scenario validations: N/a

Table 16. Test Case Id: TC_L_16_CS

Test case name	**Secure Firmware Update - Able to update firmware with ongoing transaction**	
Test case Id	TC_L_16_CS	
Use case Id(s)	L01	
Requirement(s)	L01.FR.01,L01.FR.06,L01.FR.10,L01.FR.20	
System under test	Charging Station	
Description	The CSMS is able to request the Charging Station to securely download and install a new firmware by sending an UpdateFirmwareRequest with a signingCertificate.	
Purpose	To verify if the Charging Station is able to securely download and install a new firmware, while a transaction is ongoing.	
Prerequisite(s)	- A file server has been setup according to the (by the Charging Station) supported file transfer protocol(s), indicated by the configuration variable **FileTransferProtocols**. - The Charging Station is able to update its firmware while a transaction is ongoing.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): **State is** *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **UpdateFirmwareResponse**	**1.** The OCTT sends a **UpdateFirmwareRequest** with **firmware.installDateTime** *<Current DateTime - 2 hours>* **firmware.location** *<Configured firmware_location>* **firmware.retrieveDateTime** *<Current DateTime - 2 hours>* **firmware.signingCertificate** *<Configured signingCertificate>* **firmware.signature** *<Configured signature>*
	3. The Charging Station sends a **FirmwareStatusNotificationRequest**. With **status** *Downloading*	**4.** The OCTT responds with a **FirmwareStatusNotificationResponse**.
	5. The Charging Station sends a **FirmwareStatusNotificationRequest**. With **status** *Downloaded*	**6.** The OCTT responds with a **FirmwareStatusNotificationResponse**.
	7. The Charging Station sends a **FirmwareStatusNotificationRequest**. With **status** *SignatureVerified*	**8.** The OCTT responds with a **FirmwareStatusNotificationResponse**.
	9. The Charging Station sends a **FirmwareStatusNotificationRequest**. With **status** *Installing*	**10.** The OCTT responds with a **FirmwareStatusNotificationResponse**.
	11. The OCTT waits for the Charging Station to reconnect. Note: *The Charging Station reconnects to reestablish the protocol version handshake.*	
	12. The Charging Station sends a **FirmwareStatusNotificationRequest**. With **status** *Installed*	**13.** The OCTT responds with a **FirmwareStatusNotificationResponse**.
	14. The Charging Station sends a **SecurityEventNotificationRequest** With **type** *FirmwareUpdated*	**15.** The OCTT responds with a **SecurityEventNotificationResponse**

Test case name	Secure Firmware Update - Able to update firmware with ongoing transaction
Tool validations	* Step 2: Message **UpdateFirmwareResponse** - **status** *Accepted* * Step 3: Message **FirmwareStatusNotificationRequest** - **status** *Downloading* * Step 5: Message **FirmwareStatusNotificationRequest** - **status** *Downloaded* * Step 7: Message **FirmwareStatusNotificationRequest** - **status** *SignatureVerified* * Step 9: Message **FirmwareStatusNotificationRequest** - **status** *Installing* * Step 12: Message **FirmwareStatusNotificationRequest** - **status** *Installed* * Step 14: Message **SecurityEventNotificationRequest** - **type** *FirmwareUpdated*
	Post scenario validations: N/a

Table 17. Reusable State: RebootBeforeFirmwareInstallation

State	**RebootBeforeFirmwareInstallation**	
System under test	Charging Station	
Description	The Charging Station needs to reboot before firmware installation.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The Charging Station sends a **FirmwareStatusNotificationRequest** With **status** *InstallRebooting*	**2.** The OCTT responds with a **FirmwareStatusNotificationResponse**
	Note: The steps 3 through 8 are only executed if the bootloader is able to communicate OCPP.	
	3. The Charging Station sends a **BootNotificationRequest**	**4.** The OCTT responds with a **BootNotificationResponse** with **status** *Accepted*
	5. The Charging Station notifies the CSMS about the current state of all connectors.	**6.** The OCTT responds accordingly.
	7. The Charging Station sends a **FirmwareStatusNotificationRequest** With **status** *Installing*	**8.** The OCTT responds with a **FirmwareStatusNotificationResponse**
Tool validations	* Step 1: Message **FirmwareStatusNotificationRequest** - **status** *InstallRebooting* * Step 3: Message **BootNotificationRequest** - **reason** *FirmwareUpdate* * Step 7: Message **FirmwareStatusNotificationRequest** - **status** *Installing*	
	Post scenario validations: N/a	

Table 18. Reusable State: RebootBeforeFirmwareActivation

State	**RebootBeforeFirmwareActivation**	
System under test	Charging Station	
Description	The Charging Station needs to reboot before firmware activation.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The Charging Station sends a **FirmwareStatusNotificationRequest** With **status** *InstallRebooting* Note(s): *- This step is optional. However it is recommended to notify the CSMS before rebooting the Charging Station to activate the new firmware.*	**2.** The OCTT responds with a **FirmwareStatusNotificationResponse**
	3. The Charging Station sends a **BootNotificationRequest**	**4.** The OCTT responds with a **BootNotificationResponse** with **status** *Accepted*
	5. The Charging Station notifies the CSMS about the current state of all connectors.	**6.** The OCTT responds accordingly.
Tool validations	* Step 1: Message **FirmwareStatusNotificationRequest** - **status** *InstallRebooting* * Step 3: Message **BootNotificationRequest** - **reason** *FirmwareUpdate*	
	Post scenario validations: N/a	

6.1.25. Page 241 - (2023-12) - TC_L_05_CS - Added main step and tool validation for SecurityEventNotification *InvalidFirmwareSigningCertificate*

During additional testing it was noticed that this testcase should also have been expecting a SecurityEventNotification of type *InvalidFirmwareSigningCertificate*, in accordance with requirement L01.FR.02.

Additionally, the testcase now uses a generated invalid certificate, instead of the tester needing the configure one, to improve the ease of use of the OCTT.

Added main steps

Added	**3.** The Charging Station sends a **SecurityEventNotificationRequest**. With **type** *InvalidFirmwareSigningCertificate* **4.** The OCTT responds with a **SecurityEventNotificationResponse**.

Added tool validation

Added	* Step 3: Message **SecurityEventNotificationRequest** - **type** *InvalidFirmwareSigningCertificate*

6.1.26. Page 268-281 - (2023-12) - TC_M_XX_CS - Testcases only applicable when security profile 2 or 3 is supported

These testcases are only applicable for Charging Stations that support either security profile 2 or 3. However for a Charging Station that supports only security profile 1, part 2 specification describes the following:

> - The Unsecured Transport with Basic Authentication Profile does not include authentication for the CSMS, or measures to set up a secure communication channel. Therefore, it should only be used in trusted networks, for instance in networks where there is a VPN between the CSMS and the Charging Station. For field operation it is highly recommended to use a security profile with TLS.
> - In some cases (e.g. lab installations, test setups, etc.) one might prefer to use OCPP 2.0.1 without implementing security. While this is possible, it is NOT considered a valid OCPP 2.0.1 implementation.

Therefore these testcases is mandatory to pass for certification.

Added	**Prerequisite(s)**	- The Charging Station supports Security Profile 2 or 3.

6.1.27. Page 269/276 - (2023-12) - TC_M_02_CS & TC_M_13_CS & TC_M_17_CS & TC_M_18_CS - Only applicable when signed firmware update is supported

This testcase is only applicable for Charging Stations that support **signed** firmware updates. However it is highly recommended to support the **signed** variant, opposed to the **unsigned** firmware update variant. For certification only the implementation of the **signed** firmware update is allowed, so therefore this testcase is mandatory for certification.

Additionally the existing prerequisite from TC_M_02_CS is removed, because the variable **AdditionalRootCertificateCheck** does not effect the **ManufacturerRootCertificate**.

Old text	**Prerequisite(s)**	The Charging Station does NOT have the following configuration; **AdditionalRootCertificateCheck** is implemented with value *true*
New text	**Prerequisite(s)**	- The Charging Station supports **signed** firmware updates.

6.1.28. Page 282 - (2023-12) - TC_M_23_CS - Testcase only applicable when security profile 3 is supported

This testcase is only applicable for Charging Stations that support security profile 3.

Old text	**Prerequisite(s)**	N/a
New text	**Prerequisite(s)**	- The Charging Station supports Security Profile 3. - A valid *CSMSRootCertificate* is installed on the Charging Station.

6.1.29. Page 284 - (2023-12) - TC_N_26_CS - Require a minimal size for the configured retry interval, based on the upload speed

The OCTT can is very flexible in its configurations, however some testcases prevent certain configured value combinations or require a minimal size, depending on the speed of the system. This is to prevent false positives or negatives.

Changed preparations

Old text	**Configuration State:**	N/a
New text	**Configuration State:**	The retry interval should be configured longer than the time it takes to attempt an upload.

Additionally an issue has been fixed regarding tool validation step numbering and the amount of times the OCTT expects the Charging Station to repeat step(s) (3) 5.

Changed note main step

Old text	Note(s): *- Steps 3 & 4 are optional after the first attempt.* *- The Charging Station will perform step (3,) 5, three times with <Configured retryInterval> seconds in between.*
New text	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.*

Changed tool validation step

Old text	* Step 1: Message **GetLogResponse** - **status** *Accepted*
New text	* Step 2: Message **GetLogResponse** - **status** *Accepted*

6.1.30. Page 293 - (2023-12) - TC_N_36_CS - Missing prerequisite

This testcase is only applicable for Charging Station that support cancelling an ongoing log file upload.

Added	**Prerequisite(s)**	The Charging Station supports cancelling an ongoing log file upload.

6.1.31. Page 292/293 - (2023-12) - TC_N_35_CS & TC_N_36_CS - Invalid prerequisite

Log file upload is part of functional block N, but is not related to monitoring.

Removed	**Prerequisite(s)**	Charging Station supports Monitoring

6.1.32. Page 308 - (2023-12) - Reusable State: EnergyTransferSuspended - Increased flexibility to support Charging Stations with high level communication

In case of high level communication, the transaction might already be not authorized anymore. Therefore the EnergyTransferSuspended reusable state has been made more flexible in its validations.

Changed tool validation step

Old text	* Step 1: Message: **TransactionEventRequest** - **triggerReason** must be *ChargingStateChanged* - **transactionInfo.chargingState** must be *EVConnected* OR - **transactionInfo.chargingState** must be *SuspendedEV* AND - **transactionInfo.stoppedReason** must be *StoppedByEV* - **eventType** must be *Ended* OR *Updated*
New text	* Step 1: Message: **TransactionEventRequest** - **triggerReason** must be *ChargingStateChanged* (If **chargingState** = *SuspendedEV*) - **transactionInfo.chargingState** must be *EVConnected* OR *SuspendedEV* - **transactionInfo.stoppedReason** must be *StoppedByEV* (if **eventType** = *Ended*) - **eventType** must be *Ended* OR *Updated*

6.2. Test Cases Charging Station Management System

6.2.1. Page 380 - (2023-12) - TC_E_39_CSMS - Missing requirement reference

Added applicable requirement reference.

Old text	**Requirement(s)**	E03.FR.05, E06.FR.04
New text	**Requirement(s)**	E03.FR.04, E03.FR.05, E06.FR.04

6.2.2. Page 384 - (2023-12) - TC_E_21_CSMS - Missing requirement reference

Added applicable requirement reference.

Old text	**Requirement(s)**	E06.FR.03,F03.FR.01,F03.FR.09
New text	**Requirement(s)**	E06.FR.03,F03.FR.01,F03.FR.09 , F03.FR.10

6.2.3. Page 400 - (2023-12) - TC_E_31_CSMS - Added missing StatusNotification steps

To correctly simulate the scenario, the OCTT needs to send StatusNotificationRequest(s).

Added main steps

Added	**3.** The OCTT sends a **StatusNotificationRequest** With **evseId** is *<Configured evseId>* **connectorId** is *<Configured connectorId>* **connectorStatus** is *Available* **4.** The CSMS responds with a **StatusNotificationResponse**

6.2.4. Page 407 - (2023-12) - TC_F_04_CSMS - Missing requirement reference

Added applicable requirement reference.

Old text	**Requirement(s)**	E03.FR.05
New text	**Requirement(s)**	E03.FR.04, E03.FR.05

6.2.5. Page 447 - (2023-12) - TC_L_05_CSMS - Added missing SecurityEventNotification steps

To correctly simulate the scenario, the OCTT needs to send a SecurityEventNotificationRequest.

Added main steps

Added	**3.** The OCTT sends a **SecurityEventNotificationRequest** With **type** is *InvalidFirmwareSigningCertificate* **4.** The CSMS responds with a **SecurityEventNotificationResponse**

6.2.6. Page 448 - (2023-12) - TC_L_06_CSMS - Added missing SecurityEventNotification steps

To correctly simulate the scenario, the OCTT needs to send a SecurityEventNotificationRequest.

Added main steps

Added	**9.** The OCTT sends a **SecurityEventNotificationRequest** With **type** is *InvalidFirmwareSignature* **10.** The CSMS responds with a **SecurityEventNotificationResponse**

6.2.7. Page 473 - (2023-12) - TC_E_32_CSMS - Added missing NotifyCustomerInformation steps

To correctly simulate the scenario, the OCTT needs to send a NotifyCustomerInformationRequest.

Added main steps

Added	**3.** The OCTT sends a **NotifyCustomerInformationRequest** **4.** The CSMS responds with a **NotifyCustomerInformationResponse**