OCPP 2.0.1

Part 6 - Test Cases

Edition 3 FINAL, 2024-05-06

Table of Contents

Version History

Version	Date	Modified by	Description
OCPP 2.0.1 Edition 3	2024-05-06	Open Charge Alliance	OCPP 2.0.1 Edition 3. All errata from OCPP 2.0.1 Part 6 until and including Errata 2024-04 have been merged into this version of the specification. In this edition all certification profiles are available.
1.0	2023-06-30	Open Charge Alliance	Release for Core & Advanced Security

1. Introduction

1.1. About this document

This document is created to describe a set of valid test cases for OCPP 2.0.1. These test cases can be executed using the OCPP Compliance Testing Tool (OCTT) for OCPP 2.0.1. The scenarios in the tool are described in detail including the expected behaviour of the System Under Test (SUT). This document is divided in chapters, each describing an OCPP functional block as can be found in the official OCPP specification. These are:

- A. Security
- B. Provisioning
- C. Authorization
- D. Local Authorization List Management
- E. Transactions
- F. Remote Control
- G. Availability
- H. Reservation
- I. Tariff and Cost
- J. Meter Values
- K. Smart Charging
- L. Firmware Management
- M. ISO 15118 Certificate Management
- N. Diagnostics
- O. Display Message
- P. Data Transfer

The scenarios in this document are also part of the OCA certification process of OCPP. Please refer to OCPP 2.0.1 Part 5 - Certification Profiles for more information about the relation between certification profiles and the test scenarios in this document.

1.2. Conventions

The following conventions / rules apply to all test cases, unless explicitly mentioned otherwise. These will not be mentioned separately at every test case.

- The OCPP specification is always leading.
- This document does not specify which tests need to be passed for certification, this will be specified in a separate document.
- All messages shall comply with the OCPP 2.0.1 schemas from the OCPP specification.
- The messages are to be sent as mentioned in the scenario details.
- Validations will be mentioned and grouped per step.
- Messages, datatypes and configuration variables will convey to the following formatting rules:
    - Datatypes, messages and configuration variables are displayed bold.
    - Values are displayed italic.

2. Test Cases Charging Station

2.1. General pre conditions & tool validations

General conditions/validations are overruled by testcase specific conditions/validations, unless specifically stated otherwise.

General pre conditions:

- Charging Station is Accepted by the CSMS
- Charging Station has a stable active connection to the CSMS
- Charging Station connectors are available
- Charging Station is Idle, with no active transactions
- Charging Station is clear of faults
- Charging Station has no charging schedules active
- Charging Station has no active reservations
- The Configuration variable **AuthCtrlr.LocalPreAuthorize** is set to *false*.
- Charging Station has no more OCPP messages to be send in queue
- Charging Station is not busy with transfer of diagnostics
- Charging Station is not busy with download of firmware
- Charging Station is not upgrading firmware
- Charging Station is ready to accept/start a charging session
- Charging Station has no Display message configured
- Charging Station has no active custom monitors

General tool rules/validations:

- TransactionEventRequest messages don’t have to be sent in chronological order. However the provided seqNo are sequentially numbered in chronological order. This way the CSMS is able to determine whether all messages of a transaction have been received.
- After connecting/disconnecting the EV and EVSE, the Charging Station SHALL report the new status of its connector and report any queued TransactionEventRequest(s). These message are allowed to be sent in any order.
- If the transaction was authorized with **Reusable State** *Authorized remote, then the first TransactionEventRequest sent after receiving a **RequestStartTransactionRequest** message will contain **triggerReason** with value _RemoteStart* (This will overrule the step specific tool validations) AND will contain **transactionInfo.remoteStartId**
- The first **TransactionEventRequest** of a transaction MUST contain **eventType** *Started*.
- The first **TransactionEventRequest** sent after connecting the EVSE and EV MUST contain **evse.id** and **evse.connectorId**
- The first **TransactionEventRequest** sent after presenting the idToken MUST contain **idToken** with value *<Configured valid idToken fields>*
- If the energy transfer was stopped with **Reusable State** *StopAuthorized local, then the _stoppedReason* of the last **TransactionEventRequest** of that transaction with **eventType** *Ended*, must have value *Local* OR be omitted.
- When validating/comparing time / dateTime values, the OCTT will in most cases accept a configurable deviation. The certification labs will configure a deviation of 4 seconds.
- Every FirmwareStatusNotificationRequest sent for a firmware update SHALL contain the same requestId as the UpdateFirmwareRequest that started the firmware update.
- The list of ChargingSchedulePeriod elements in a chargingSchedule SHALL be ordered by increasing values of ChargingSchedulePeriod.startPeriod. This means the list is in chronological order.
- 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.

2.2. A Security

Table 1. Test Case Id: TC_A_01_CS

Test case name	**Basic Authentication - Valid username/password combination**	
Test case Id	TC_A_01_CS	
Use case Id(s)	A00, B01	
Requirement(s)	A00.FR.202, A00.FR.203, A00.FR.204, A00.FR.205, A00.FR.301, A00.FR.302, A00.FR.304 AND B01.FR.01, B01.FR.05, B01.FR.09	
System under test	Charging Station	
Description	The Charging Station uses Basic authentication to authenticate itself to the CSMS, when using security profile 1 or 2.	
Purpose	To verify whether the Charging Station is able to authenticate itself to the CSMS using Basic Authentication.	
Prerequisite(s)	- The charging station supports security profile 1 and/or 2 - The active NetworkConnectionProfile uses either security profile 1 OR 2.	
Before (Preparations)	**Configuration State:** **SecurityCtrlr.BasicAuthPassword** is *<Configured basicAuthPassword>*	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *Booted*	
Tool validations	* Step 1: The authorization header of the HTPP upgrade request must be formatted as follows: *AUTHORIZATION: Basic <Base64 encoded(<ChargingStationId>:<Configured basicAuthPassword>)>* - The ChargingStationId, must equal the ChargingStationId provided at the end of the connection url string of the HTTP request. - BasicAuthPassword must consist of minimum 16 and maximum 40 characters - BasicAuthPassword may only contain alpha-numeric characters and the special characters allowed by identifierString.	
	Post scenario validations: N/a	

Table 2. Test Case Id: TC_A_04_CS

Test case name	**TLS - server-side certificate - Valid certificate**	
Test case Id	TC_A_04_CS	
Use case Id(s)	A00	
Requirement(s)	A00.FR.309,A00.FR.312,A00.FR.313,A00.FR.319,A00.FR.321,A00.FR.412,A00.FR.422	
System under test	Charging Station	
Description	The CSMS uses a server-side certificate to identify itself to the Charging Station, when using security profile 2 or 3.	
Purpose	To verify whether the Charging Station is able to receive a server certificate provided by the CSMS and setup a secured WebSocket connection.	
Prerequisite(s)	- The charging station supports security profile 2 and/or 3 - The active NetworkConnectionProfile uses either security profile 2 OR 3.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): **State is** *Booting*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The Charging Station initiates a TLS handshake and sends a Client Hello to the OCTT.	**2.** The OCTT responds with a Server Hello With the <Configured server certificate>
	3. The Charging Station performs the following actions: Send client certificate Client Key Exchange Certificate verify Change Cipher Spec Finished Note(s): *- The client certificate is only sent when the Charging Station uses security profile 3.*	**4.** The OCTT performs the following actions: Change Cipher Spec Finished
	5. The Charging Station sends a HTTP upgrade request to the OCTT Note(s): *- The HTTP request only contains a username/password combination when the Charging Station uses security profile 2.*	**6.** The OCTT upgrades the connection to a (secured) WebSocket connection.
	7. The Charging Station sends a **BootNotificationRequest**	**8.** The OCTT responds with a **BootNotificationResponse** with **status** *Accepted*
	9. The Charging Station notifies the CSMS about the current state of all connectors.	**10.** The OCTT responds accordingly.

Test case name	**TLS - server-side certificate - Valid certificate**
Tool validations	* Step 2: The OCTT validates the following before sending the server certificate: - The Charging Station must use TLS version 1.2 or above At least the following set of cipher suites must be supported: (TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 AND TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384) OR (TLS_RSA_WITH_AES_128_GCM_SHA256 AND TLS_RSA_WITH_AES_256_GCM_SHA384) * Step 9: Message: **StatusNotificationRequest** - **connectorStatus** *Available* Message: **NotifyEventRequest** - **eventData[0].trigger** *Delta* - **eventData[0].actualValue** *"Available"* - **eventData[0].component.name** *"Connector"* - **eventData[0].variable.name** *"AvailabilityState"*
	Post scenario validations: N/a

Table 3. Test Case Id: TC_A_05_CS

Test case name	**TLS - server-side certificate - Invalid certificate**	
Test case Id	TC_A_05_CS	
Use case Id(s)	A00	
Requirement(s)	A00.FR.309,A00.FR.310,A00.FR.311,A00.FR.412,A00.FR.413,A00.FR.414	
System under test	Charging Station	
Description	The CSMS uses a server-side certificate to identify itself to the Charging Station, when using security profile 2 or 3.	
Purpose	To verify whether the Charging Station is able to terminate the connection when the received server certificate is invalid.	
Prerequisite(s)	- The charging station supports security profile 2 and/or 3 - The active NetworkConnectionProfile uses either security profile 2 OR 3. - This testcase can be executed multiple times, using different kinds of invalid certificates: Unknown certificate expired certificate certificate with commonName that does not equal the FQDN of the CSMS.	
Before (Preparations)	**Configuration State:** **OCPPCommCtrlr.NetworkProfileConnectionAttempts** is *2*	
	Memory State: N/a	
	Reusable State(s): **State is** *Booting*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The Charging Station initiates a TLS handshake and sends a Client Hello to the OCTT.	**2.** The OCTT responds with a Server Hello With a <Configured invalid server certificate>
	3. The Charging Station deems the server certificate invalid and terminates the connection.	
	4. The Charging Station initiates a TLS handshake and sends a Client Hello to the OCTT.	**5.** The OCTT responds with a Server Hello With the <Configured server certificate>

Test case name	**TLS - server-side certificate - Invalid certificate**	
	6. The Charging Station performs the following actions: Send client certificate Client Key Exchange Certificate verify Change Cipher Spec Finished Note(s): *- The client certificate is only sent when the Charging Station uses security profile 3.*	**7.** The OCTT performs the following actions: Change Cipher Spec Finished
	8. The Charging Station sends a HTTP upgrade request to the OCTT Note(s): *- The HTTP request only contains a username/password combination when the Charging Station uses security profile 2.*	**9.** The OCTT upgrades the connection to a (secured) WebSocket connection.
	10. The Charging Station sends a **BootNotificationRequest**	**11.** The OCTT responds with a **BootNotificationResponse** with **status** *Accepted*
	12. The Charging Station notifies the CSMS about the current state of all connectors.	**13.** The OCTT responds accordingly.
	14 The Charging Station sends a **SecurityEventNotificationRequest**	**15** The OCTT responds with a **SecurityEventNotificationResponse**
Tool validations	* Step 14: Message: **SecurityEventNotificationRequest** - **type** must be *InvalidCsmsCertificate*	
	Post scenario validations: N/a	

Table 4. Test Case Id: TC_A_06_CS

Test case name	**TLS - server-side certificate - TLS version too low**	
Test case Id	TC_A_06_CS	
Use case Id(s)	A00	
Requirement(s)	A00.FR.314,A00.FR.316,A00.FR.416,A00.FR.417,A00.FR.419	
System under test	Charging Station	
Description	The CSMS uses a server-side certificate to identify itself to the Charging Station, when using security profile 2 or 3.	
Purpose	To verify whether the Charging Station is able to terminate the connection when it notices the used TLS version is lower than 1.2.	
Prerequisite(s)	- The charging station supports security profile 2 and/or 3 - The active NetworkConnectionProfile uses either security profile 2 OR 3.	
Before (Preparations)	**Configuration State:** **OCPPCommCtrlr.NetworkProfileConnectionAttempts** is *1*	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The Charging Station initiates a TLS handshake and sends a Client Hello to the OCTT.	**2.** The OCTT responds with a Server Hello, but uses a TLS version lower than 1.2 With a <Configured server certificate>
	3. The Charging Station notices the used TLS version is lower than 1.2 and terminates the connection.	
	4. The Charging Station initiates a TLS handshake and sends a Client Hello to the OCTT.	**5.** The OCTT responds with a Server Hello With the <Configured server certificate>

Test case name	**TLS - server-side certificate - TLS version too low**	
	6. The Charging Station performs the following actions: Send client certificate Client Key Exchange Certificate verify Change Cipher Spec Finished Note(s): *- The client certificate is only sent when the Charging Station uses security profile 3.*	**7.** The OCTT performs the following actions: Change Cipher Spec Finished
	8. The Charging Station sends a HTTP upgrade request to the OCTT Note(s): *- The HTTP request only contains a username/password combination when the Charging Station uses security profile 2.*	**9.** The OCTT upgrades the connection to a (secured) WebSocket connection.
	10. The Charging Station sends a **BootNotificationRequest**	**11.** The OCTT responds with a **BootNotificationResponse** with **status** *Accepted*
	12. The Charging Station notifies the CSMS about the current state of all connectors.	**13.** The OCTT responds accordingly.
	14 The Charging Station sends a **SecurityEventNotificationRequest**	**15** The OCTT responds with a **SecurityEventNotificationResponse**
	16 The Charging Station sends a **SecurityEventNotificationRequest**	**17** The OCTT responds with a **SecurityEventNotificationResponse**
	Note(s): *- The order in which the requests of steps 12 and 14 and 16 arrive is not relevant.* *- Steps 16 and 17 are optional as the Charging Station might not be able to detect that the TLS handshake failed, because of invalid TLS version.*	
Tool validations	* Step 14: Message: **SecurityEventNotificationRequest** - **type** must be *StartupOfTheDevice* or *ResetOrReboot*	
	* Step 16: Message: **SecurityEventNotificationRequest** - **type** must be *InvalidTLSVersion*	

Table 5. Test Case Id: TC_A_07_CS

Test case name	**TLS - Client-side certificate - valid certificate**	
Test case Id	TC_A_07_CS	
Use case Id(s)	A00	
Requirement(s)	A00.FR.401,A00.FR.402,A00.FR.415,A00.FR.416,A00.FR.422,A00.FR.502,A00.FR.503,A00.FR.507,A00.FR.508,A00.FR.511	
System under test	Charging Station	
Description	The Charging Station uses a client-side certificate to identify itself to the CSMS, when using security profile 3.	
Purpose	To verify whether the Charging Station is able to provide a valid client certificate and setup a secured WebSocket connection.	
Prerequisite(s)	- The charging station supports security profile 3 - The active NetworkConnectionProfile uses security profile 3.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): **State is** *Booting*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The Charging Station initiates a TLS handshake and sends a Client Hello to the OCTT.	**2.** The OCTT responds with a Server Hello With the <Configured server certificate>
	3. The Charging Station performs the following actions: Send client certificate Client Key Exchange Certificate verify Change Cipher Spec Finished	**4.** The OCTT performs the following actions: Change Cipher Spec Finished
	5. The Charging Station sends a HTTP upgrade request to the OCTT	**6.** The OCTT upgrades the connection to a (secured) WebSocket connection.
	7. The Charging Station sends a **BootNotificationRequest**	**8.** The OCTT responds with a **BootNotificationResponse** with **status** *Accepted*
	9. The Charging Station notifies the CSMS about the current state of all connectors.	**10.** The OCTT responds accordingly.

Test case name	**TLS - Client-side certificate - valid certificate**
Tool validations	* Step 4: The OCTT validates the following before finishing the TLS handshake: - The Charging Station must use TLS version 1.2 or above At least the following set of cipher suites must be supported: (TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 AND TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384) OR (TLS_RSA_WITH_AES_128_GCM_SHA256 AND TLS_RSA_WITH_AES_256_GCM_SHA384) - When using RSA or DSA the key must be at least 2048 bits long. and when using elliptic curve cryptography the key must be at least 224 bits long. - The received Client side certificate must be transmitted in the X.509 format encoded in Privacy-Enhanced Mail (PEM) format. - The certificate must include a serial number. - The subject field of the certificate must contain a commonName RDN which consists of the unique serial number of the Charging Station. *NOTE: If one of the above validations fails, the OCTT can still setup the WebSocket connection (if it is able to), but the testcase will FAIL and the OCTT reports why it failed.* * Step 9: Message: **StatusNotificationRequest** - **connectorStatus** *Available* Message: **NotifyEventRequest** - **eventData[0].trigger** *Delta* - **eventData[0].actualValue** *"Available"* - **eventData[0].component.name** *"Connector"* - **eventData[0].variable.name** *"AvailabilityState"*
	Post scenario validations: N/a

Table 6. Test Case Id: TC_A_09_CS

Test case name	**Update Charging Station Password for HTTP Basic Authentication - Accepted**	
Test case Id	TC_A_09_CS	
Use case Id(s)	A01	
Requirement(s)	A01.FR.01, A01.FR.11, A01.FR.12, B01.FR.01	
System under test	Charging Station	
Description	This test case defines how to use the BasicAuthPassword, the password used to authenticate Charging Stations in security profile 1 (Basic Authentication) and security profile 2 (TLS with Basic Authentication)	
Purpose	To verify if the Charging Station is able to accept and store and log the new BasicAuthPassword as described at the OCPP specification.	
Prerequisite(s)	The charging station supports security profile 1 and/or 2	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SetVariablesResponse**	**1.** The OCTT sends a **SetVariablesRequest** with **setVariableData**[1]: - **variable.name** = *"BasicAuthPassword"* - **component.name** = *"SecurityCtrlr"* - **attributeValue** = *"<NewPassword>"*
	3. The ChargingStation sends a HTTP upgrade request with an Authorization header, containing a username/password combination (with the new *BasicAuthPassword*). Note(s): *- The Authorization header is formatted as follows: AUTHORIZATION: Basic <Base64 encoded(<Configured ChargingStationId>:<NEW BasicAuthPassword>)>*	**4.** The OCTT validates the username/password combination AND upgrades the connection to a (secured) WebSocket connection.
	5. The Charging Station sends a **BootNotificationRequest**	**6.** The OCTT responds with a **BootNotificationResponse**
	7. The Charging Station notifies the OCTT about the current state of all connectors.	**8.** The OCTT responds accordingly.
	Note(s): *- Steps 5, 6, 7, and 8 are only required when **status** in Step 2 is RebootRequired*	
Tool validations	* Step 2: Message: **SetVariablesResponse** - **status** must be *Accepted* or *RebootRequired*	
	Post scenario validations: N/a	

Table 7. Test Case Id: TC_A_10_CS

Test case name	**Update Charging Station Password for HTTP Basic Authentication - Rejected**	
Test case Id	TC_A_10_CS	
Use case Id(s)	A01	
Requirement(s)	A01.FR.01, A01.FR.11, A01.FR.12	
System under test	Charging Station	
Description	This test case defines how to use the BasicAuthPassword, the password used to authenticate Charging Stations in security profile 1 (Basic Authentication) and security profile 2 (TLS with Basic Authentication)	
Purpose	To verify if the Charging Station is able to reject the new BasicAuthPassword.	
Prerequisite(s)	The charging station supports security profile 1 and/or 2	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SetVariablesResponse**	**1.** The OCTT sends a **SetVariablesRequest** **setVariableData**[1]: - **variable.name** = *"BasicAuthPassword"* - **component.name** = *"SecurityCtrlr"* - **attributeValue** = *"<NewPassword which is less than 16 characters>"*
	3. The ChargingStation sends a HTTP upgrade request with an Authorization header, containing a username/password combination (with the old *BasicAuthPassword*). Note(s): *- The Authorization header is formatted as follows: AUTHORIZATION: Basic <Base64 encoded(<Configured ChargingStationId>:<OLD BasicAuthPassword>)>*	**4.** The OCTT validates the username/password combination AND upgrades the connection to a (secured) WebSocket connection.
	5. Execute **Reusable State** *Booted*	
Tool validations	* Step 2: Message: **SetVariablesResponse** - **status** must be *Rejected*	
	Post scenario validations: BasicAuthPassword should be *<Configured BasicAuthPassword>* N/a	

Table 8. Test Case Id: TC_A_11_CS

Test case name	**Update Charging Station Certificate by request of CSMS - Success - Charging Station Certificate**	
Test case Id	TC_A_11_CS	
Use case Id(s)	A02 & F06	
Requirement(s)	A02.FR.02, A02.FR.03, A02.FR.06, A02.FR.08, A02.FR.09 & F06.FR.04,F06.FR.05,F06.FR.10	
System under test	Charging Station	
Description	The CSMS is able to request the Charging Station to update its charging station certificate using the TriggerMessageRequest message.	
Purpose	To verify if the Charging Station is able to update its Charging Station Certificate.	
Prerequisite(s)	- The charging station supports security profile 3 - The active NetworkConnectionProfile uses security profile 3.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *RenewChargingStationCertificate* for certificateType *ChargingStationCertificate*	
Tool validations	N/a	
	Post scenario validations: N/a	

Table 9. Test Case Id: TC_A_12_CS

Test case name	**Update Charging Station Certificate by request of CSMS - Success - V2G Certificate**	
Test case Id	TC_A_12_CS	
Use case Id(s)	A02 & F06	
Requirement(s)	A02.FR.02, A02.FR.03, A02.FR.06,A02.FR.13,A02.FR.15 & F06.FR.04,F06.FR.05,F06.FR.10	
System under test	Charging Station	
Description	The CSMS is able to request the Charging Station to update its charging station certificate using the TriggerMessageRequest message.	
Purpose	To verify if the Charging Station is able to update its V2G Charging Station Certificate.	
Prerequisite(s)	The Charging Station supports ISO 15118.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Memory State** *RenewV2GChargingStationCertificate*	
Tool validations	N/a	
	Post scenario validations: N/a	

Table 10. Test Case Id: TC_A_13_CS

Test case name	**Update Charging Station Certificate by request of CSMS - Success - Combined Certificate**	
Test case Id	TC_A_13_CS	
Use case Id(s)	A00, A02	
Requirement(s)	A00.FR.428,A02.FR.02, A02.FR.03, A02.FR.06	
System under test	Charging Station	
Description	The CSMS is able to request the Charging Station to update its charging station certificate using the TriggerMessageRequest message.	
Purpose	To verify if the Charging Station is able to update its combined V2G / Charging Station Certificate.	
Prerequisite(s)	- The Charging Station supports security profile 3 - The active NetworkConnectionProfile uses security profile 3. - The Charging Station supports ISO 15118.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **TriggerMessageResponse**	**1.** The OCTT sends a **TriggerMessageRequest** With **requestedMessage** *SignCombinedCertificate*
	3 The Charging Station sends a **SignCertificateRequest**	**4.** The OCTT responds with a **SignCertificateResponse** With **status** *Accepted*
	6. The Charging Station responds with a **CertificateSignedResponse**	**5.** The OCTT sends a **CertificateSignedRequest** With **certificateChain** *<Certificate generated from the received CSR from step 3 and signed by the V2GRoot OR SubCA certificate from the configured V2G certificate chain>*
Tool validations	* Step 2: Message: **TriggerMessageResponse** - **status** must be *Accepted* * Step 3: Message: **SignCertificateRequest** - **csr** must contain *<An CSR that meets the following requirements:* *The key must be at least 224 bits long.* *The received CSR must be transmitted as described in RFC 2986 and then encoded in Privacy-Enhanced Mail (PEM) format.>* * Step 6: Message: **CertificateSignedResponse** - **status** must be *Accepted*	
	Post scenario validations: N/a	

Table 11. Test Case Id: TC_A_14_CS

Test case name	**Update Charging Station Certificate by request of CSMS - Invalid certificate**	
Test case Id	TC_A_14_CS	
Use case Id(s)	A02	
Requirement(s)	A02.FR.07,A03.FR.07	
System under test	Charging Station	
Description	The CSMS is able to request the Charging Station to update its charging station certificate using the TriggerMessageRequest message.	
Purpose	To verify if the Charging Station is able to discard an invalid certificate and report a security event.	
Prerequisite(s)	- The Charging Station supports security profile 3 - The active NetworkConnectionProfile uses security profile 3.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **TriggerMessageResponse**	**1.** The OCTT sends a **TriggerMessageRequest** With **requestedMessage** *SignChargingStationCertificate*
	3 The Charging Station sends a **SignCertificateRequest**	**4.** The OCTT responds with a **SignCertificateResponse** With **status** *Accepted*
	6. The Charging Station responds with a **CertificateSignedResponse**	**5.** The OCTT sends a **CertificateSignedRequest** With **certificateChain** *<Configured invalid_signingCertificate>* **certificateType** *ChargingStationCertificate*
	7 The Charging Station sends a **SecurityEventNotificationRequest**	**8** The OCTT responds with a **SecurityEventNotificationResponse**
Tool validations	* Step 2: Message: **TriggerMessageResponse** - **status** must be *Accepted* * Step 3: Message: **SignCertificateRequest** - **csr** must contain *<An CSR that meets the following requirements:* *When using RSA or DSA the key must be at least 2048 bits long.* *and when using elliptic curve cryptography the key must be at least 224 bits long.* *The received CSR must be transmitted as described in RFC 2986 and then encoded in Privacy-Enhanced Mail (PEM) format.>* * Step 6: Message: **CertificateSignedResponse** - **status** must be *Rejected* * Step 7: Message: **SecurityEventNotificationRequest** - **type** must be *InvalidChargingStationCertificate*	
	Post scenario validations: N/a	

Table 12. Test Case Id: TC_A_15_CS

Test case name	**Update Charging Station Certificate by request of CSMS - SignCertificateRequest Rejected**	
Test case Id	TC_A_15_CS	
Use case Id(s)	A02	
Requirement(s)	N/a	
System under test	Charging Station	
Description	The CSMS is able to request the Charging Station to update its charging station certificate using the TriggerMessageRequest message.	
Purpose	To verify if the Charging Station is able to discard an invalid certificate and report a security event.	
Prerequisite(s)	- The Charging Station supports security profile 3 - The active NetworkConnectionProfile uses security profile 3.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **TriggerMessageResponse**	**1.** The OCTT sends a **TriggerMessageRequest** With **requestedMessage** *SignChargingStationCertificate*
	3 The Charging Station sends a **SignCertificateRequest**	**4.** The OCTT responds with a **SignCertificateResponse** With **status** *Rejected*
Tool validations	* Step 2: Message: **TriggerMessageResponse** - **status** must be *Accepted*	
	Post scenario validations: N/a	

Table 13. Test Case Id: TC_A_23_CS

Test case name	**Update Charging Station Certificate by request of CSMS - CertificateSignedRequest Timeout**	
Test case Id	TC_A_23_CS	
Use case Id(s)	A02 & F06	
Requirement(s)	A02.FR.17,A02.FR.18	
System under test	Charging Station	
Description	The CSMS is able to request the Charging Station to update its charging station certificate using the TriggerMessageRequest message.	
Purpose	To verify if the Charging Station is able to send a new signCertificateRequest when it did not receive a certificateSignedRequest after the configured timeout.	
Prerequisite(s)	- The charging station supports security profile 3 - The Charging Station supports the CertificateSignedRequest Timeout feature	
Before (Preparations)	**Configuration State:** **SecurityCtrlr.CertSigningWaitMinimum** is *<Configured CertSigningWaitMinimum>* **SecurityCtrlr.CertSigningRepeatTimes** is *1*	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **TriggerMessageResponse**	**1.** The OCTT sends a **TriggerMessageRequest** With **requestedMessage** *SignChargingStationCertificate*
	3 The Charging Station sends a **SignCertificateRequest**	**4.** The OCTT responds with a **SignCertificateResponse** With **status** *Accepted*
		5. The OCTT does NOT send the **CertificateSignedRequest** and waits for the **SignCertificateRequest** to be resend after the *<Configured CertSigningWaitMinimum>*
	6 The Charging Station sends a **SignCertificateRequest**	**7.** The OCTT responds with a **SignCertificateResponse** With **status** *Accepted*
		8. The OCTT does NOT send the **CertificateSignedRequest** and waits for the **SignCertificateRequest** to be resend after the *<Configured CertSigningWaitMinimum>* times 2
	9 The Charging Station sends a **SignCertificateRequest**	**10.** The OCTT responds with a **SignCertificateResponse** With **status** *Accepted*
	12. The Charging Station responds with a **CertificateSignedResponse**	**11.** The OCTT sends a **CertificateSignedRequest** With **certificateChain** *<Certificate generated from the received CSR from step 3 and signed by the provided CSMS Root certificate>* **certificateType** *ChargingStationCertificate*

Test case name	**Update Charging Station Certificate by request of CSMS - CertificateSignedRequest Timeout**
Tool validations	* Step 2: Message: **TriggerMessageResponse** - **status** must be *Accepted* * Step 3/6/9: Message: **SignCertificateRequest** - **csr** must contain *<An CSR that meets the following requirements:* *When using RSA or DSA the key must be at least 2048 bits long.* *and when using elliptic curve cryptography the key must be at least 224 bits long.* *The received CSR must be transmitted as described in RFC 2986 and then encoded in Privacy-Enhanced Mail (PEM) format.>* * Step 5: - The Charging Station shall not resend the **SignCertificateRequest** before the *<Configured CertSigningWaitMinimum>* expired * Step 8: - The Charging Station shall not resend the **SignCertificateRequest** before the *<Configured CertSigningWaitMinimum>* times 2 expired * Step 12: Message: **CertificateSignedResponse** - **status** must be *Accepted*
	Post scenario validations: N/a

Table 14. Test Case Id: TC_A_19_CS

Test case name	**Upgrade Charging Station Security Profile - Accepted**	
Test case Id	TC_A_19_CS	
Use case Id(s)	A05	
Requirement(s)	A05.FR.04,A05.FR.05,A05.FR.06	
System under test	Charging Station	
Description	The CSMS updates the connection details on the Charging Station, to increase the security profile level.	
Purpose	To verify if the Charging Station is able to increase the security profile level when configured to do so by the CSMS.	
Prerequisite(s)	Security profile must be set to 1 or 2	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: If configured <Security profile> is 1, then *CertificateInstalled* If configured <Security profile> is 2, then *RenewChargingStationCertificate*	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SetNetworkProfileResponse**	**1.** The OCTT sends a **SetNetworkProfileRequest** with **configurationSlot** is *<Configured configurationSlot>* or *<Configured configurationSlot>* depending on which one is already in use - **connectionData.messageTimeout** *<Configured messageTimeout>* - **connectionData.ocppCsmsUrl** *<Configured ocppCsmsUrl>* - **connectionData.ocppInterface** *<Configured ocppInterface>* - **connectionData.ocppVersion** *OCPP20* - **connectionData.securityProfile** *<Configured securityProfile + 1>*
	4. The Charging Station responds with a **SetVariablesResponse**	**3.** The OCTT sends a **SetVariablesRequest** with **variable.name** is *"NetworkConfigurationPriority"* **component.name** is *"OCPPCommCtrlr"* **attributeValue** is *"<Configured configurationSlot + 1>,<Configured configurationSlot>"*
	6. The Charging Station responds with a **ResetResponse**	**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.*
	7. The Charging Station reconnects to the OCTT with security profile is <Configured securityProfile + 1>	**8.** The OCTT accepts the connection attempt.
	9. Execute **Reusable State** *Booted*	
	11. The Charging Station responds with **GetVariablesResponse**	**10.** OCTT sends **GetVariablesRequest** with: - **variable.name** = *"SecurityProfile"* - **component.name** = *"SecurityCtrlr"*
	13. The Charging Station responds with **GetVariablesResponse**	**12.** OCTT sends **GetVariablesRequest** with: - **variable.name** = *"NetworkConfigurationPriority"* - **component.name** = *"OCPPCommCtrlr"*

Test case name	**Upgrade Charging Station Security Profile - Accepted**
Tool validations	* Step 2: Message **SetNetworkProfileResponse** - **status** *Accepted* * Step 4: Message **SetVariablesResponse** - **setVariableResult[0].attributeStatus** *Accepted* OR *RebootRequired* * Step 6: Message **ResetResponse** - **status** *Accepted* * Step 11: Message **GetVariablesResponse** - **getVariableResult[0].attributeValue** *<Configured securityProfile + 1>* * Step 13: Message **GetVariablesResponse** - **getVariableResult[0].attributeValue** Does not contain <Configured configurationSlot>
	Post scenario validations: - N/a

Table 15. Test Case Id: TC_A_20_CS

Test case name	**Upgrade Charging Station Security Profile - No valid CSMSRootCertificate installed**	
Test case Id	TC_A_20_CS	
Use case Id(s)	A05	
Requirement(s)	A05.FR.02	
System under test	Charging Station	
Description	The CSMS is able to change the connectionData at the Charging Station. By doing this it is able to upgrade the connection to a higher security profile.	
Purpose	To verify if the Charging Station is able to reject upgrading to a higher security profile when it does not have a valid CSMSRootCertificate installed.	
Prerequisite(s)	- The OCTT connectionData configuration for SUT Charging Station only allows for ip addresses the OCTT is able to bind. - The Charging Station support at least 2 security profiles, one of which is security profile 1. - The Charging Station does not have a valid CSMSRootCertificate installed. - The first OCTT connectionData configuration slot must be configured for security profile 1. - The second OCTT connectionData configuration slot must be configured for security profile 2 or 3. - When starting this testcase the OCTT will start another webSocket server for the second connectionData slot.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SetNetworkProfileResponse**	**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>*
	4. The Charging Station responds with a **SetVariablesResponse**	**3.** The OCTT sends a **SetVariablesRequest** with **variable.name** is *"NetworkConfigurationPriority"* **component.name** is *"OCPPCommCtrlr"* **attributeValue** is *<Configured configurationSlot2>,<Configured configurationSlot>*
Tool validations	* Step 2: Message **SetNetworkProfileResponse** - **status** *Accepted* * Step 4: Message **SetVariablesResponse** - **setVariableResult[0].attributeStatus** *Rejected*	
	Post scenario validations: - N/a	

Table 16. Test Case Id: TC_A_21_CS

Test case name	**Upgrade Charging Station Security Profile - No valid ChargingStationCertificate installed**	
Test case Id	TC_A_21_CS	
Use case Id(s)	A05	
Requirement(s)	A05.FR.03	
System under test	Charging Station	
Description	The CSMS is able to change the connectionData at the Charging Station. By doing this it is able to upgrade the connection to a higher security profile.	
Purpose	To verify if the Charging Station is able to reject upgrading to a higher security profile when it does not have a valid ChargingStationCertificate installed.	
Prerequisite(s)	- The OCTT connectionData configuration for SUT Charging Station only allows for ip addresses the OCTT is able to bind. - The Charging Station support at least 2 security profiles. - The Charging Station does not have a valid ChargingStationCertificate installed. - The Charging Station has a valid CSMSRootCertificate installed. - The second OCTT connectionData configuration slot must be configured for security profile 3. - When starting this testcase the OCTT will start another webSocket server for the second connectionData slot.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SetNetworkProfileResponse**	**1.** The OCTT sends a **SetNetworkProfileRequest** with - **configurationSlot** is *<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>*
	4. The Charging Station responds with a **SetVariablesResponse**	**3.** The OCTT sends a **SetVariablesRequest** with **variable.name** is *"NetworkConfigurationPriority"* **component.name** is *"OCPPCommCtrlr"* **attributeValue** is *<Configured configurationSlot2>,<Configured configurationSlot>*
Tool validations	* Step 2: Message **SetNetworkProfileResponse** - **status** *Accepted* * Step 4: Message **SetVariablesResponse** - **setVariableResult[0].attributeStatus** *Rejected*	
	Post scenario validations: - N/a	

Table 17. Test Case Id: TC_A_22_CS

Test case name	**Upgrade Charging Station Security Profile - Downgrade security profile - Rejected**	
Test case Id	TC_A_22_CS	
Use case Id(s)	A05, B09	
Requirement(s)	B09.FR.04	
System under test	Charging Station	
Description	The CSMS is able to change the connectionData at the Charging Station. It tries to downgrade the connection to a lower security profile.	
Purpose	To verify if the Charging Station is able to reject downgrading to a lower security profile than the currently active security profile.	
Prerequisite(s)	- The OCTT connectionData configuration for SUT Charging Station only allows for ip addresses the OCTT is able to bind. - The Charging Station supports security profile 2 and/or 3. - The second OCTT connectionData configuration slot must be configured for a security profile lower than the first OCTT connectionData configuration slot. - When starting this testcase the OCTT will start another webSocket server for the second connectionData slot.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SetNetworkProfileResponse**	**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>*
Tool validations	* Step 2: Message **SetNetworkProfileResponse** - **status** *Rejected*	
	Post scenario validations: - N/a	

2.3. B Provisioning

Table 18. Test Case Id: TC_B_01_CS

Test case name	**Cold Boot Charging Station - Accepted**	
Test case Id	TC_B_01_CS	
Use case Id(s)	B01	
Requirement(s)	B01.FR.01, B01.FR.05, B01.FR.09	
System under test	Charging Station	
Description	The booting mechanism allows a Charging Station to provide some general information about the Charging Station to the CSMS on startup AND it allows the Charging Station to request whether it is allowed to start sending other OCPP messages.	
Purpose	To verify whether the Charging Station is able to perform the booting mechanism as described at the OCPP specification.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *Booted*	
Tool validations	N/a	
	Post scenario validations: - A message to report the state of a connector has been received for all connectors.	

Table 19. Test Case Id: TC_B_02_CS

Test case name	**Cold Boot Charging Station - Pending**	
Test case Id	TC_B_02_CS	
Use case Id(s)	B02, F06	
Requirement(s)	B02.FR.01, B02.FR.02, B02.FR.04, B02.FR.05, B02.FR.06, B02.FR.08, F06.FR.17	
System under test	Charging Station	
Description	The booting mechanism allows a Charging Station to provide some general information about the Charging Station to the CSMS on startup AND it allows the Charging Station to request whether it is allowed to start sending other OCPP messages. A CSMS can temporarily halt the Charging Stations operations by returning the Pending status at the BootNotificationResponse. During this time the CSMS is able to retrieve and set configurations from the Charging Station.	
Purpose	To verify whether the Charging Station is able to correctly handle the pending state of the boot mechanism.	
Prerequisite(s)	The testcases; TC_B_06_CS, TC_B_09_CS, TC_B_13_CS are executed with test result *PASS*.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	<u>Manual Action</u>: *Reboot the Charging Station.*	
	1. The Charging Station sends a **BootNotificationRequest**	**2.** The OCTT responds with a **BootNotificationResponse** with **status** *Pending* **interval** *<Configured heartbeatInterval>*
	4. The Charging Station responds with **SetVariablesResponse**	**3.** OCTT sends **SetVariablesRequest** with: - **variable.name** = *"OfflineThreshold"* - **component.name** = *"OCPPCommCtrlr"* - **attributeValue** = *"300"* - **attributeType** is omitted
	6. The Charging Station responds with **GetVariablesResponse**	**5.** OCTT sends **GetVariablesRequest** with: - **variable.name** = *"OfflineThreshold"* - **component.name** = *"OCPPCommCtrlr"* - **attributeType** is omitted
	8. Charging Station responds with: **GetBaseReportResponse**	**7.** OCTT sends **GetBaseReportRequest** with: - **requestId** = *<Generated requestId>* - **reportBase** = *FullInventory*

Test case name	**Cold Boot Charging Station - Pending**	
	Charging Station	**CSMS**
	9. Charging Station responds with: **NotifyReportRequest** Note(s): *- This step is repeated as often as needed to report all configuration variables.*	**10.** OCTT sends **NotifyReportResponse**
	12. The Charging Station responds with a **RequestStartTransactionResponse**	**11.** The OCTT sends a **RequestStartTransactionRequest** Note(s): *- This step is executed after the OCTT received all NotifyReport messages. This is indicated by the tbc and seqNo fields.*
	14. The Charging Station responds with a **TriggerMessageResponse**	**13.** The OCTT sends a **TriggerMessageRequest** with **requestedMessage** *BootNotification*
	15. The Charging Station sends a **BootNotificationRequest** Note(s): *- The Charging Station resends the BootNotificationRequest after having responded to the TriggerMessageRequest, so before the interval from the BootNotificationResponse has been passed.*	**16.** The OCTT responds with a **BootNotificationResponse** with **status** *Accepted* **interval** *<Configured heartbeatInterval>*
	17. The Charging Station notifies the CSMS about the current state of all connectors.	**18.** The OCTT responds accordingly.

Test case name	**Cold Boot Charging Station - Pending**
Tool validations	* Step 4: Message: **SetVariablesResponse** - **setVariableResult[0].attributeStatus** *Accepted* * Step 6: Message: **GetVariablesResponse** - **getVariableResult[0].attributeStatus** *Accepted* * Step 8: Message: **GetBaseReportResponse** - **status** *Accepted* * Step 12: Message: **RequestStartTransactionResponse** - **status** *Rejected* * Step 14: Message: **TriggerMessageResponse** - **status** *Accepted* or *NotImplemented* * Step 15: Message: **BootNotificationRequest** - **reason** *Triggered* (If the **status** from the response from step 14 contained *Accepted*) * Step 17: Message: **StatusNotificationRequest** - **connectorStatus** *Available* Message: **NotifyEventRequest** - **eventData[0].trigger** *Delta* - **eventData[0].actualValue** *"Available"* - **eventData[0].component.name** *"Connector"* - **eventData[0].variable.name** *"AvailabilityState"*
	Post scenario validations: - A message to report the state of a connector has been received for all connectors.

Table 20. Test Case Id: TC_B_03_CS

Test case name	**Cold Boot Charging Station - Rejected**	
Test case Id	TC_B_03_CS	
Use case Id(s)	B03	
Requirement(s)	B03.FR.02, B03.FR.04, B03.FR.06	
System under test	Charging Station	
Description	The booting mechanism allows a Charging Station to provide some general information about the Charging Station to the CSMS on startup AND it allows the Charging Station to request whether it is allowed to start sending other OCPP messages.	
Purpose	To verify whether the Charging Station is able to correctly handle a rejected BootNotification.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Reboot the Charging Station.*	
	1. The Charging Station sends a **BootNotificationRequest**	**2.** The OCTT responds with a **BootNotificationResponse** with **status** *Rejected* **interval** *<Configured heartbeatInterval>*
	3. The Charging Station sends a **BootNotificationRequest** Note(s): *- The Charging Station resends the BootNotificationRequest after x seconds, whereby x is equal to or greater than the interval from the BootNotificationResponse.* *- The Charging Station is not allowed to send any OCPP message in the meantime.* *- The Charging Station is allowed to close the connection until it needs to resend the BootNotificationRequest.*	**4.** The OCTT responds with a **BootNotificationResponse** with **status** *Accepted* **interval** *<Configured heartbeatInterval>*
	5. The Charging Station notifies the CSMS about the current state of all connectors.	**6.** The OCTT responds accordingly.
Tool validations	* Step 5: Message: **StatusNotificationRequest** - **connectorStatus** *Available* Message: **NotifyEventRequest** - **eventData[0].trigger** *Delta* - **eventData[0].actualValue** *"Available"* - **eventData[0].component.name** *"Connector"* - **eventData[0].variable.name** *"AvailabilityState"*	
	Post scenario validations: - A message to report the state of a connector has been received for all connectors.	

Table 21. Test Case Id: TC_B_30_CS

Test case name	**Cold Boot Charging Station - Pending/Rejected - SecurityError**	
Test case Id	TC_B_30_CS	
Use case Id(s)	B03	
Requirement(s)	B03.FR.08	
System under test	Charging Station	
Description	The booting mechanism allows a Charging Station to provide some general information about the Charging Station to the CSMS on startup AND it allows the Charging Station to request whether it is allowed to start sending other OCPP messages. The CSMS may respond to the BootNotificationRequest with status *Rejected*. During this state, the Charging Station is not allowed to send RPC Framework: CALL message that is NOT a BootNotificationRequest.	
Purpose	To verify whether the Charging Station is able to handle unauthorized messages from the CSMS by responding with a SecurityError.	
Prerequisite(s)		
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The Charging Station sends a **BootNotificationRequest**	**2.** The OCTT responds with a **BootNotificationResponse** with **status** *Rejected*
	4. The Charging Station responds with RPC Framework: CALLERROR: SecurityError.	**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
Tool validations	N/a	
	N/a	

Table 22. Test Case Id: TC_B_06_CS

Test case name	**Get Variables - single value**	
Test case Id	TC_B_06_CS	
Use case Id(s)	B06	
Requirement(s)	B06.FR.01, B06.FR.02, B06.FR.03, B06.FR.04, B06.FR.10, B06.FR.11	
System under test	Charging Station	
Description	Get the value of one of the required variables of OCPPCommCtrlr	
Purpose	To test getting a single value using GetVariablesRequest for one of the mandatory component/variable combinations that must exist in the DM implementation.	
Prerequisite(s)	N/A	
Before (Preparations)	**Configuration State:** **OCPPCommCtrlr.OfflineThreshold** is *300*	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. Charging Station responds with **GetVariablesResponse**	**1.** OCTT sends **GetVariablesRequest** with: - **variable.name** = *"OfflineThreshold"* - **component.name** = *"OCPPCommCtrlr"* - **attributeType** = *Actual*
Tool validations	* Step 2: Message: **GetVariablesResponse** - **attributeStatus** = *Accepted* - **attributeType** = *Actual* - **attributeValue** = *"300"* - **component.name** = *"OCPPCommCtrlr"* - **variable.name** = *"OfflineThreshold"* - **attributeStatusInfo** is absent or **attributeStatusInfo.reasonCode** = *"NoError"*	
	Post scenario validations: N/A	

Table 23. Test Case Id: TC_B_07_CS

Test case name	**Get Variables - multiple values**	
Test case Id	TC_B_07_CS	
Use case Id(s)	B06	
Requirement(s)	B06.FR.01, B06.FR.02, B06.FR.03, B06.FR.04, B06.FR.10	
System under test	Charging Station	
Description	Get the value of two required variables	
Purpose	To test getting multiple values using GetVariablesRequest for one of the mandatory component/variable combinations that must exist in the DM implementation.	
Prerequisite(s)	N/A	
Before (Preparations)	**Configuration State:** **OCPPCommCtrlr.OfflineThreshold** is *300* **AuthCtrlr.LocalAuthorizeOffline** is *true*	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. Charging Station responds with **GetVariablesResponse** with **attributeStatus** = *Accepted*.	**1.** OCTT sends **GetVariablesRequest** with: - **getVariableData[0].variable.name** = *"OfflineThreshold"* - **getVariableData[0].component.name** = *"OCPPCommCtrlr"* - **getVariableData[0].attributeType** = *Actual* - **getVariableData[1].variable.name** = *"LocalAuthorizeOffline"* - **getVariableData[1].component.name** = *"AuthCtrlr"* - **getVariableData[1].attributeType** = *Actual*
Tool validations	* Step 2: Message: **GetVariablesResponse** has (in arbitrary order) **GetVariableResultType**[0]: - **attributeStatus** = *Accepted* - **attributeType** = *Actual* - **attributeValue** = *300* - **component.name** = *"OCPPCommCtrlr"* - **variable.name** = *"OfflineThreshold"* - **attributeStatusInfo** is absent or **attributeStatusInfo.reasonCode** = *"NoError"* **GetVariableResultType**[1]: - **attributeStatus** = *Accepted* - **attributeType** = *Actual* - **attributeValue** = *"true"* - **component.name** = *"AuthCtrlr"* - **variable.name** = *"LocalAuthorizeOffline"* - **attributeStatusInfo** is absent or **attributeStatusInfo.reasonCode** = *"NoError"*	
	Post scenario validations: N/A	

Table 24. Test Case Id: TC_B_32_CS

Test case name	**Get Variables - Unknown component**	
Test case Id	TC_B_32_CS	
Use case Id(s)	B06	
Requirement(s)	B06.FR.06	
System under test	Charging Station	
Description	The CSMS can use a GetVariablesRequest to retrieve values from device model variables at the Charging Station.	
Purpose	To verify whether the Charging Station can handle receiving a GetVariablesRequest for an unknown component.	
Prerequisite(s)	N/A	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with **GetVariablesResponse**	**1.** OCTT sends **GetVariablesRequest** with: - **variable.name** = *"OfflineThreshold"* - **component.name** = *"UnknownComponent"* - **attributeType** is omitted
Tool validations	* Step 2: Message: **GetVariablesResponse** - **getVariableResult[0].attributeStatus** = *UnknownComponent* - **getVariableResult[0].component.name** = *"UnknownComponent"* - **getVariableResult[0].variable.name** = *"OfflineThreshold"*	
	Post scenario validations: N/A	

Table 25. Test Case Id: TC_B_33_CS

Test case name	**Get Variables - Unknown variable**	
Test case Id	TC_B_33_CS	
Use case Id(s)	B06	
Requirement(s)	B06.FR.07	
System under test	Charging Station	
Description	The CSMS can use a GetVariablesRequest to retrieve values from device model variables at the Charging Station.	
Purpose	To verify whether the Charging Station can handle receiving a GetVariablesRequest for an unknown variable.	
Prerequisite(s)	N/A	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with **GetVariablesResponse**	**1.** OCTT sends **GetVariablesRequest** with: - **variable.name** = *"UnknownVariable"* - **component.name** = *"OCPPCommCtrlr"* - **attributeType** is omitted
Tool validations	* Step 2: Message: **GetVariablesResponse** - **getVariableResult[0].attributeStatus** = *UnknownVariable* - **getVariableResult[0].component.name** = *"OCPPCommCtrlr"* - **getVariableResult[0].variable.name** = *"UnknownVariable"*	
	Post scenario validations: N/A	

Table 26. Test Case Id: TC_B_34_CS

Test case name	**Get Variables - Not supported attribute type**	
Test case Id	TC_B_34_CS	
Use case Id(s)	B06	
Requirement(s)	B06.FR.08	
System under test	Charging Station	
Description	The CSMS can use a GetVariablesRequest to retrieve values from device model variables at the Charging Station.	
Purpose	To verify whether the Charging Station can handle receiving a GetVariablesRequest for a not supported attribute type.	
Prerequisite(s)	N/A	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with **GetVariablesResponse**	**1.** OCTT sends **GetVariablesRequest** with: - **variable.name** = *"OfflineThreshold"* - **component.name** = *"OCPPCommCtrlr"* - **attributeType** = *Target*
Tool validations	* Step 2: Message: **GetVariablesResponse** - **getVariableResult[0].attributeStatus** = *NotSupportedAttributeType* - **getVariableResult[0].component.name** = *"OCPPCommCtrlr"* - **getVariableResult[0].variable.name** = *"OfflineThreshold"* - **getVariableResult[0].attributeType** = *Target*	
	Post scenario validations: N/A	

Table 27. Test Case Id: TC_B_09_CS

Test case name	**Set Variables - single value**	
Test case Id	TC_B_09_CS	
Use case Id(s)	B05	
Requirement(s)	B05.FR.01, B05.FR.02, B05.FR.03, B05.FR.10, B05.FR.12	
System under test	Charging Station	
Description	Set the value of one of the required variables of OCPPCommCtrlr	
Purpose	To test setting a single value using SetVariablesRequest for one of the mandatory component/variable combinations that must exist in the DM implementation.	
Prerequisite(s)	N/A	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. Charging Station responds with **SetVariablesResponse** with **attributeStatus** = *Accepted*.	**1.** OCTT sends **SetVariablesRequest** with: - **variable.name** = *"OfflineThreshold"* - **component.name** = *"OCPPCommCtrlr"* - **attributeValue** = *"300"* - **attributeType** *Actual*
Tool validations	* Step 2: Message: **SetVariablesResponse** - **setVariableResult[0].attributeStatus** = *Accepted* - **setVariableResult[0].attributeType** = *Actual* - **setVariableResult[0].component.name** = *"OCPPCommCtrlr"* - **setVariableResult[0].variable.name** = *"OfflineThreshold"* - **setVariableResult[0].attributeStatusInfo** is absent or **setVariableResult[0].attributeStatusInfo.reasonCode** = *"NoError"*	
	Post scenario validations: N/A	

Table 28. Test Case Id: TC_B_10_CS

Test case name	**Set Variables - multiple values**	
Test case Id	TC_B_10_CS	
Use case Id(s)	B05	
Requirement(s)	B05.FR.01, B05.FR.02, B05.FR.03, B05.FR.10, B05.FR.12	
System under test	Charging Station	
Description	Set the value of two required variables	
Purpose	To test setting multiple values using GetVariablesRequest for one of the mandatory component/variable combinations that must exist in the DM implementation.	
Prerequisite(s)	N/A	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. Charging Station responds with **SetVariablesResponse** with **attributeStatus** = *Accepted*.	**1.** OCTT sends **SetVariablesRequest** with: - **setVariableData[0].variable.name** = *"OfflineThreshold"* - **setVariableData[0].component.name** = *"OCPPCommCtrlr"* - **setVariableData[0].attributeValue** = *"300"* - **setVariableData[0].attributeType** = *Actual* - **setVariableData[1].variable.name** = *"LocalAuthorizeOffline"* - **setVariableData[1].component.name** = *"AuthCtrlr"* - **setVariableData[1].attributeValue** = *"true"* - **setVariableData[0].attributeType** = *Actual*
Tool validations	* Step 2: Message: **SetVariablesResponse** has (in arbitrary order) **SetVariableResultType**[1]: - **attributeStatus** = *Accepted* - **attributeType** = *Actual* - **component.name** = *"OCPPCommCtrlr"* - **variable.name** = *"OfflineThreshold"* - **attributeStatusInfo** is absent or **attributeStatusInfo.reasonCode** = *"NoError"* **SetVariableResultType**[2]: - **attributeStatus** = *Accepted* - **attributeType** = *Actual* - **component.name** = *"AuthCtrlr"* - **variable.name** = *"LocalAuthorizeOffline"* - **attributeStatusInfo** is absent or **attributeStatusInfo.reasonCode** = *"NoError"*	
	Post scenario validations: N/A	

Table 29. Test Case Id: TC_B_35_CS

Test case name	**Set Variables - Unknown component**	
Test case Id	TC_B_35_CS	
Use case Id(s)	B05	
Requirement(s)	B05.FR.04	
System under test	Charging Station	
Description	The CSMS can use a SetVariablesRequest to set values from device model variables at the Charging Station.	
Purpose	To verify whether the Charging Station can handle receiving a SetVariablesRequest for an unknown component.	
Prerequisite(s)	N/A	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with **SetVariablesResponse**	**1.** OCTT sends **SetVariablesRequest** with: - **variable.name** = *"OfflineThreshold"* - **component.name** = *"UnknownComponent"* - **attributeType** is omitted
Tool validations	* Step 2: Message: **SetVariablesResponse** - **setVariableResult[0].attributeStatus** = *UnknownComponent* - **setVariableResult[0].component.name** = *"UnknownComponent"* - **setVariableResult[0].variable.name** = *"OfflineThreshold"*	
	Post scenario validations: N/A	

Table 30. Test Case Id: TC_B_36_CS

Test case name	**Set Variables - Unknown variable**	
Test case Id	TC_B_36_CS	
Use case Id(s)	B05	
Requirement(s)	B05.FR.05	
System under test	Charging Station	
Description	The CSMS can use a SetVariablesRequest to set values from device model variables at the Charging Station.	
Purpose	To verify whether the Charging Station can handle receiving a SetVariablesRequest for an unknown variable.	
Prerequisite(s)	N/A	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with **SetVariablesResponse**	**1.** OCTT sends **SetVariablesRequest** with: - **variable.name** = *"UnknownVariable"* - **component.name** = *"OCPPCommCtrlr"* - **attributeType** is omitted
Tool validations	* Step 2: Message: **SetVariablesResponse** - **setVariableResult[0].attributeStatus** = *UnknownVariable* - **setVariableResult[0].component.name** = *"OCPPCommCtrlr"* - **setVariableResult[0].variable.name** = *"UnknownVariable"*	
	Post scenario validations: N/A	

Table 31. Test Case Id: TC_B_37_CS

Test case name	**Set Variables - Not supported attribute type**	
Test case Id	TC_B_37_CS	
Use case Id(s)	B05	
Requirement(s)	B05.FR.06	
System under test	Charging Station	
Description	The CSMS can use a SetVariablesRequest to set values from device model variables at the Charging Station.	
Purpose	To verify whether the Charging Station can handle receiving a SetVariablesRequest for a not supported attribute type.	
Prerequisite(s)	N/A	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with **SetVariablesResponse**	**1.** OCTT sends **SetVariablesRequest** with: - **variable.name** = *"OfflineThreshold"* - **component.name** = *"OCPPCommCtrlr"* - **attributeType** = *Target*
Tool validations	* Step 2: Message: **SetVariablesResponse** - **setVariableResult[0].attributeStatus** = *NotSupportedAttributeType* - **setVariableResult[0].component.name** = *"OCPPCommCtrlr"* - **setVariableResult[0].variable.name** = *"OfflineThreshold"* - **setVariableResult[0].attributeType** = *Target*	
	Post scenario validations: N/A	

Table 32. Test Case Id: TC_B_11_CS

Test case name	**Set Variables - invalidly formatted values**
Test case Id	TC_B_11_CS
Use case Id(s)	B05
Requirement(s)	B05.FR.07
System under test	Charging Station
Description	Set the value of two of the required variables of OCPPCommCtrlr
Purpose	To test setting of variables of different type with invalidly formatted values using GetVariablesRequest for one of the mandatory component/variable combinations that must exist in the DM implementation.
Prerequisite(s)	Charging Station DM has the variable "NextTimeOffsetTransitionDateTime" of component "ClockCtrlr" to test setting of a date.
Before (Preparations)	**Configuration State:** N/a
	Memory State: N/a
	Reusable State(s): N/a

Test case name	**Set Variables - invalidly formatted values**	
Main (Test scenario)	**Charging Station**	**CSMS**
	Notes: *Steps 1 to 8 are repeated 5 times for value = <configured offlineThreshold>, <configured offlineThreshold + 0.1>, true, currentTime, "abc"*	
	2. Charging Station responds with **SetVariablesResponse** with *If value not supported:* **attributeStatus** = *Rejected* **attributeStatusInfo** = *InvalidValue* *If component/variable/value supported:* **attributeStatus** = *Accepted*	**1.** OCTT sends **SetVariablesRequest** with - **variable.name** = *"OfflineThreshold"* - **component.name** = *"OCPPCommCtrlr"* - **attributeValue** = *value*
	Notes: *Steps 3 and 4 will only be tested if this component/variable combination is supported*	
	4. Charging Station responds with **SetVariablesResponse** with *If value not supported:* **attributeStatus** = *Rejected* **attributeStatusInfo** = *InvalidValue* *If component/variable/value supported:* **attributeStatus** = *Accepted*	**3.** OCTT sends **SetVariablesRequest** with - **variable.name** = *"LimitChangeSignificance"* - **component.name** = *"SmartChargingCtrlr"* - **attributeValue** = *value*
	Notes: *Steps 5 and 6 will only be executed if this component/variable combination is readwrite*	
	6. Charging Station responds with **SetVariablesResponse** with *If value not supported:* **attributeStatus** = *Rejected* **attributeStatusInfo** = *InvalidValue* *If component/variable/value supported:* **attributeStatus** = *Accepted*	**5.** OCTT sends **SetVariablesRequest** with: - **variable.name** = *"AuthorizeRemoteStart"* - **component.name** = *"AuthCtrlr"* - **attributeValue** = *value*
	Notes: *Steps 7 and 8 will only be executed if the CS supports this component/variable combination*	
	8. Charging Station responds with **SetVariablesResponse** with *If value not supported:* **attributeStatus** = *Rejected* **attributeStatusInfo** = *InvalidValue* *If component/variable/value supported:* **attributeStatus** = *Accepted*	**7.** OCTT sends **SetVariablesRequest** with: - **variable.name** = *"NextTimeOffsetTransitionDateTime"* - **component.name** = *"ClockCtrlr"* - **attributeValue** = *value*

Test case name	**Set Variables - invalidly formatted values**
Tool validations	* Step 2: Message: **SetVariablesResponse** has **SetVariableResultType** - **attributeStatus** = *Rejected/Accepted* - **attributeType** = *Actual* - **component.name** = *"OCPPCommCtrlr"* - **variable.name** = *"OfflineThreshold"* - **attributeStatusInfo** is absent or **attributeStatusInfo.reasonCode** = *InvalidValue* (not required)
	* Step 4: Message: **SetVariablesResponse** has **SetVariableResultType** - **attributeStatus** = *Rejected/Accepted* - **attributeType** = *Actual* - **component.name** = *"AuthCtrlr"* - **variable.name** = *"AuthorizeRemoteStart"* - **attributeStatusInfo** is absent or **attributeStatusInfo.reasonCode** = *InvalidValue* (not required)
	* Step 6: Message: **SetVariablesResponse** has **SetVariableResultType** - **attributeStatus** = *Rejected/Accepted* - **attributeType** = *Actual* - **component.name** = *"SmartChargingCtrlr"* - **variable.name** = *"LimitChangeSignificance"* - **attributeStatusInfo** is absent or **attributeStatusInfo.reasonCode** = *InvalidValue* (not required)
	* Step 8: Message: **SetVariablesResponse** has **SetVariableResultType** - **attributeStatus** = *Rejected/Accepted* - **attributeType** = *Actual* - **component.name** = *"ClockCtrlr"* - **variable.name** = *"NextTimeOffsetTransitionDateTime"* - **attributeStatusInfo** is absent or **attributeStatusInfo.reasonCode** = *InvalidValue* (not required)
	Post scenario validations: N/A

Table 33. Test Case Id: TC_B_39_CS

Test case name	**Set Variables - Read-only**	
Test case Id	TC_B_39_CS	
Use case Id(s)	B05	
Requirement(s)	B05.FR.09	
System under test	Charging Station	
Description	The CSMS can use a SetVariablesRequest to set values from device model variables at the Charging Station.	
Purpose	To verify whether the Charging Station can handle receiving a SetVariablesRequest for a Read-only variable.	
Prerequisite(s)	N/A	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with **SetVariablesResponse**	**1.** OCTT sends **SetVariablesRequest** with: - **variable.name** = *"MessageTimeout"* - **variable.instance** = *"Default"* - **component.name** = *"OCPPCommCtrlr"* - **attributeType** is omitted
Tool validations	* Step 2: Message: **SetVariablesResponse** - **setVariableResult[0].attributeStatus** = *Rejected* - **setVariableResult[0].component.name** = *"OCPPCommCtrlr"* - **setVariableResult[0].variable.name** = *"MessageTimeout"* - **setVariableResult[0].variable.instance** = *"Default"*	
	Post scenario validations: N/A	

Table 34. Test Case Id: TC_B_12_CS

Test case name	**Get Base Report - ConfigurationInventory**	
Test case Id	TC_B_12_CS	
Use case Id(s)	B07	
Requirement(s)	B07.FR.01, B07.FR.03, B07.FR.04, **B07.FR.07**, B07.FR.10, B07.FR.12	
System under test	Charging Station	
Description	CSMS requests a ConfigurationInventory base report.	
Purpose	To test that Charging Station supports the ConfigurationInventory base report.	
Prerequisite(s)	N/A	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. Charging Station responds with: **GetBaseReportResponse**	**1.** OCTT sends **GetBaseReportRequest** with: - **requestId** = *<Generated requestId>* - **reportBase** = *ConfigurationInventory*
	3. Charging Station responds with: **NotifyReportRequest**	**4.** OCTT sends **NotifyReportResponse**
	Step 3 and 4 are repeated as often as needed to report all configuration variables.	
Tool validations	* Step 2: Message: **GetBaseReportResponse** - **status** = *Accepted* - **statusInfo** is absent or **statusInfo.reasonCode** = *"NoError"*	
	* Step 3: Message: **NotifyReportRequest** - **requestId** = *<Generated requestId>* - **generatedAt** = *<timestamp at charging station>* - **seqNo** = *0* - if **variableCharacteristics.dataType** = *OptionList*, *SequenceList* or *MemberList* then **valuesList** must be provided.	
	while **tbc** = *true*	Expect **NotifyReportRequest** - **seqNo** is incremented by 1
	Post scenario validations: Check for all received variables: - **variableCharacteristics** are present - **mutability** = *ReadWrite* or *WriteOnly* Validate that as a minimum the required writable variables in section "Referenced Components and Variables" are reported, that are relevant to each functional block that has been implemented.	

Table 35. Test Case Id: TC_B_13_CS

Test case name	**Get Base Report - FullInventory**	
Test case Id	TC_B_13_CS	
Use case Id(s)	B07	
Requirement(s)	B07.FR.01, B07.FR.03, B07.FR.04, **B07.FR.08**, B07.FR.10, B07.FR.12	
System under test	Charging Station	
Description	CSMS requests a FullInventory base report.	
Purpose	To test that Charging Station supports the FullInventory base report.	
Prerequisite(s)	N/A	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. Charging Station responds with: **GetBaseReportResponse**	**1.** OCTT sends **GetBaseReportRequest** with: - **requestId** = *<Generated requestId>* - **reportBase** = *FullInventory*
	3. Charging Station responds with: **NotifyReportRequest**	**4.** OCTT sends **NotifyReportResponse**
	Step 3 and 4 are repeated as often as needed to report all configuration variables.	
Tool validations	* Step 2: Message: **GetBaseReportResponse** - **status** = *Accepted* - **statusInfo** is absent or **statusInfo.reasonCode** = *"NoError"*	
	* Step 3: Message: **NotifyReportRequest** - **requestId** = *<Generated requestId>* - **generatedAt** = *<timestamp at charging station>* - **seqNo** = *0* - if **variableCharacteristics.dataType** = *OptionList*, *SequenceList* or *MemberList* then **valuesList** must be provided.	
	while **tbc** = *true*	Expect **NotifyReportRequest** - **seqNo** is incremented by 1
	Post scenario validations: Check for all received variables: - **variableCharacteristics** are present Validate that as a minimum the required variables mentioned in section "Charging Infrastructure Related" are reported as well as the required variables in section "Referenced Components and Variables", that are relevant to each functional block that has been implemented.	

Table 36. Test Case Id: TC_B_14_CS

Test case name	**Get Base Report - SummaryInventory**	
Test case Id	TC_B_14_CS	
Use case Id(s)	B07	
Requirement(s)	B07.FR.01, B07.FR.03, B07.FR.04, **B07.FR.09**, B07.FR.10, B07.FR.12	
System under test	Charging Station	
Description	CSMS requests a SummaryInventory base report.	
Purpose	To test that Charging Station supports the SummaryInventory base report.	
Prerequisite(s)	Charging Station implementation supports the optional SummaryInventory report	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. Charging Station responds with: **GetBaseReportResponse**	**1.** OCTT sends **GetBaseReportRequest** with: - **requestId** = *<Generated requestId>* - **reportBase** = *SummaryInventory*
	3. Charging Station responds with: **NotifyReportRequest**	**4.** OCTT sends **NotifyReportResponse**
	Step 3 and 4 are repeated as often as needed to report all configuration variables.	
Tool validations	* Step 2: Message: **GetBaseReportResponse** - **status** = *Accepted* - **statusInfo** is absent or **statusInfo.reasonCode** = *"NoError"*	
	* Step 3: Message: **NotifyReportRequest** - **requestId** = *<Generated requestId>* - **generatedAt** = *<timestamp at charging station>* - **seqNo** = *0*	
	while **tbc** = *true*	Expect **NotifyReportRequest** - **seqNo** is incremented by 1
	Post scenario validations: Check for all received variables: - **variableCharacteristics** are present - if **variableCharacteristics.dataType** = *OptionList*, *SequenceList* or *MemberList* then **valuesList** must be provided. Result must be a report that lists Components/Variables relating to the Charging Station's current charging availability, and to any existing problem conditions. - For the Charging Station Component: **AvailabilityState** - For each EVSE Component: **AvailabilityState** - For each Connector Component: **AvailabilityState** (if known and different from EVSE). - For all Components in an abnormal State: - **Problem**, **Tripped**, **Overload**, **Fallback** variables. - Any other diagnostically relevant Variables of the Components.	

Table 37. Test Case Id: TC_B_15_CS

Test case name	**Get Base Report - Not Supported base report**	
Test case Id	TC_B_15_CS	
Use case Id(s)	B07	
Requirement(s)	B07.FR.02	
System under test	Charging Station	
Description	CSMS requests a base report that is not supported.	
Purpose	To test that Charging Station returns NotSupported when a SummaryInventory base report is requested, but Charging Station does not support it.	
Prerequisite(s)	Charging Station implementation does not support the optional SummaryInventory report.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. Charging Station responds with: **GetBaseReportResponse**	**1.** OCTT sends **GetBaseReportRequest** with: - **requestId** = *<Generated requestId>* - **reportBase** = *SummaryInventory*
	Note(s): *- OCTT waits to make sure CS does not send a NotifyReportRequest*	
Tool validations	* Step 2: Charging Station responds with: **GetBaseReportResponse** with: - **status** = *NotSupported* - **statusInfo** is absent or **statusInfo.reasonCode** = *"UnsupportedParam"*	
	Post scenario validations: N/A	

Table 38. Test Case Id: TC_B_16_CS

Test case name	**Get Custom Report - with component criteria**	
Test case Id	TC_B_16_CS	
Use case Id(s)	B08	
Requirement(s)	B08.FR.01, B08.FR.03, B08.FR.04, B08.FR.07, B08.FR.09, B089.FR.10, B08.FR.12, B08.FR.13, B08.FR.14	
System under test	Charging Station	
Description	CSMS requests a custom report based on a set of component criteria.	
Purpose	To test that Charging Station supports a custom report query.	
Prerequisite(s)	Charging Station has implemented custom reporting (use case B08).	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. Charging Station responds: **GetReportResponse**	**1.** OCTT sends **GetReportRequest** with: - **requestId** = *<Generated requestId>* - **componentCriteria** = { *Enabled* }
	3. Charging Station sends **NotifyReportRequest** with: - **reportData** for all components that have the variable "Enabled" set to "true".	**4.** OCTT responds with **NotifyReportResponse**
	Steps 3 and 4 may be repeated multiple times until everything has been reported.	
	OCTT sends a GetVariables for all variables to match results.	
	6. Charging Station responds with: **GetVariablesResponse**	**5.** OCTT sends **GetVariablesRequest**
Tool validations	* Step 2: +2. Message: **GetReportResponse** with: - **status** = *Accepted* - **statusInfo** is absent or **statusInfo** = *"NoError"*	
	* step 3: Message: **NotifyReportRequest** with: - **requestId** = *<Generated requestId>* - **generatedAt** = <time of generation at charging station> - **seqNo** = *0* - **reportData.variableCharacteristics** are present - **reportData.variable.name** is "Enabled" - **reportData.variableAttribute.value** = *"true"* *Note: for Enabled there will only be an Actual value.* *It does not make any sense to have a MinSet, MaxSet or Target value for them.*	
	While **tbc** = *true*	Message: **NotifyReportRequest** - **seqNo** is incremented by one - **reportData.variableCharacteristics** are present - **reportData.variable.name** is "Enabled" - **reportData.variableAttribute.value** = *"true"*
	* Step 6: Message: **GetVariablesResponse** with: - **attributeStatus** = *Accepted* - **attributeValue** = *true*	
	Post scenario validations: Check that every variable, named "Enabled" that was reported in the FullInventory **base report** with a value of "true" is also reported by the **custom report** for **componentCriteria** = { *Enabled* }.	

Table 39. Test Case Id: TC_B_17_CS

Test case name	**Get Custom Report - with component/variable**	
Test case Id	TC_B_17_CS	
Use case Id(s)	B08	
Requirement(s)	B08.FR.01, B08.FR.03, B08.FR.04, B08.FR.11, B08.FR.12, B08.FR.14	
System under test	Charging Station	
Description	CSMS requests a custom report for AvailabilityState of EVSE #1.	
Purpose	To test that Charging Station supports a custom report query.	
Prerequisite(s)	Charging Station has implemented custom reporting (use case B08).	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. Charging Station responds: **GetReportResponse**	**1.** OCTT sends **GetReportRequest** with: - **requestId** = *<Generated requestId>* - **componentVariable[0].component.name**= *"EVSE"* - **componentVariable[0].component.evse.id** = *1* - **componentVariable[0].variable.name** = *"AvailabilityState"*
	3. Charging Station sends **NotifyReportRequest**	**4.** OCTT responds with **NotifyReportResponse**
Tool validations	* Step 2: Message: **GetReportResponse** with: - **status** = *Accepted* - **statusInfo** is absent or **statusInfo** = *"NoError"*	
	* step 3: Message: **NotifyReportRequest** with: - **requestId** = *GetReportRequest requestid* - **generatedAt** = <time of generation at charging station> - **seqNo** = *0* - **reportData.component.name** = *"EVSE"* - **reportData.component.evse.id** = *1* - **reportData.variable.name** = *"AvailabilityState"* - **reportData.variableCharacteristics.dataType** = *OptionList* - **reportData.variableCharacteristics.valuesList** = *"Available, Occupied, Reserved, Unavailable, Faulted"* - **reportData.variableAttribute.mutability** = *ReadOnly* - **reportData.variableAttribute.type** = *Actual* *Note: for AvailabilityState there will only be an _Actual* value. It does not make any sense to have a *MinSet*, *MaxSet* or *Target* value for it._	
	Post scenario validations: N/A	

Table 40. Test Case Id: TC_B_18_CS

Test case name	**Get Custom Report - with component criteria and component/variable**	
Test case Id	TC_B_18_CS	
Use case Id(s)	B08	
Requirement(s)	B08.FR.01, B08.FR.03, B08.FR.04, B08.FR.05, B08.FR.11, B08.FR.12, B08.FR.14, B08.FR.15	
System under test	Charging Station	
Description	CSMS requests a custom report for AvailabilityState of EVSE #1 as *Available* and with *Problem*.	
Purpose	To test that Charging Station supports a custom report query and that it takes the component criteria into account, by first request a selection that should return a value and then requesting a selection that should not return a value.	
Prerequisite(s)	Charging Station has implemented custom reporting (use case B08).	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	Request EVSE::AvailabilityState from _Available components. Note 1: EVSE #1 must be available, because a Charging Station without EVSE is useless. Note 2: Do not confuse *Available* with *AvailabilityState*._	
	2. Charging Station responds: **GetReportResponse**	**1.** OCTT sends **GetReportRequest** with: - **requestId** = *<Generated requestId>* - **componentCriteria** = { *Available* } - **componentVariable[0].component.name** = *"EVSE"* - **componentVariable[0].component.evse.id** = *1* - **componentVariable[0].variable.name** = *"AvailabilityState"*
	3. Charging Station sends **NotifyReportRequest**	**4.** OCTT responds with **NotifyReportResponse**
	Request EVSE::AvailabilityState from _Problem components. Note 1: Assuming EVSE #1 does not have *Problem* variable set._	
	6. Charging Station responds: **GetReportResponse**	**5.** OCTT sends **GetReportRequest** with: - **requestId** = *<Generated requestId>* - **componentCriteria[0]** = *Problem* - **componentVariable[0].component.name** = *"EVSE"* - **componentVariable[0].component.evse.id** = *1* - **componentVariable[0].variable.name** = *"AvailabilityState"*

Test case name	**Get Custom Report - with component criteria and component/variable**
Tool validations	* Step 2: Message: **GetReportResponse** with: - **status** = *Accepted* - **statusInfo** is absent or **statusInfo** = *"NoError"*
	* step 3: Message: **NotifyReportRequest** with: - **requestId** = *GetReportRequest requestid* - **generatedAt** = <time of generation at charging station> - **seqNo** = *0* - **reportData.component.name** = *"EVSE"* - **reportData.component.evse.id** = *1* - **reportData.variable.name** = *"AvailabilityState"* - **reportData.variableCharacteristics.dataType** = *OptionList* - **reportData.variableCharacteristics.valuesList** = *"Available, Occupied, Reserved, Unavailable, Faulted"* - **reportData.variableAttribute.mutability** = *ReadOnly* - **reportData.variableAttribute.type** = *Actual* *Note: for AvailabilityState there will only be an _Actual* value. It does not make any sense to have a *MinSet*, *MaxSet* or *Target* value for it._
	* Step 6: Message: **GetReportResponse** with: - **status** = *EmptyResultSet* - **statusInfo** is absent or **statusInfo.reasonCode** = *"NotFound"*
	Post scenario validations: N/A

NOTE	*Test Case Id: TC_B_19_CS* Since ComponentCriterionEnum is defined as enumeration, this will most likely already be caught by the JSON parser.

Test case name	**Get Custom Report - for unknown component criteria**	
Test case Id	TC_B_19_CS	
Use case Id(s)	B08	
Requirement(s)	B08.FR.02	
System under test	Charging Station	
Description	CSMS sends a GetReport with an invalid value in **componentCriteria**.	
Purpose	To test that Charging Station returns a *NotSupported* return code in response to an invalid value for **componentCriteria**.	
Prerequisite(s)	The Charging Station has one or more not supported componentCriteria.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. Charging Station responds with: **GetReportResponse**	**1.** OCTT sends **GetReportRequest** with: - **requestId** = *<Generated requestId* - **componentCriteria = { *Available, <Configured Unsupported componentCriteria>* }** **- *componentVariable** is absent
Tool validations	* Step 2 Message: **GetReportResponse** - **status** = *NotSupported* - **statusInfo** is absent or **statusInfo.reasonCode** = *"UnsupportedParam"* or **statusInfo.reasonCode** = *"InvalidValue"*	
	Post scenario validations: N/A	

Table 41. Test Case Id: TC_B_20_CS

Test case name	**Reset Charging Station - Without ongoing transaction - OnIdle**	
Test case Id	TC_B_20_CS	
Use case Id(s)	B11	
Requirement(s)	B11.FR.01, B11.FR.03, B11.FR.04, B01.FR.03	
System under test	Charging Station	
Description	This test case covers how the CSMS can request the Charging Station to reset itself by sending a ResetRequest without any ongoing transaction. This could for example be necessary if the Charging Station is not functioning correctly.	
Purpose	To verify if the Charging Station is able to perform the reset mechanism as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **ResetResponse**	**1.** The OCTT sends a **ResetRequest** with **type** *OnIdle*
	<u>Note(s)</u>: - *Charging Station reboots*	
	3. The Charging Station sends a **BootNotificationRequest**	**4.** The OCTT responds with a **BootNotificationResponse** with **status** *Accepted*
	5. The Charging Station notifies the CSMS about the current state of all connectors.	**6.** The OCTT responds accordingly.
Tool validations	* Step 2: Message **ResetResponse** - **status** *Accepted* * Step 5: Message: **StatusNotificationRequest** - **connectorStatus** *Available* Message: **NotifyEventRequest** - **eventData[0].trigger** *Delta* - **eventData[0].actualValue** *"Available"* - **eventData[0].component.name** *"Connector"* - **eventData[0].variable.name** *"AvailabilityState"*	
	Post scenario validations: - A message to report the state of a connector has been received for all connectors.	

Table 42. Test Case Id: TC_B_21_CS

Test case name	**Reset Charging Station - With Ongoing Transaction - OnIdle**	
Test case Id	TC_B_21_CS	
Use case Id(s)	B12	
Requirement(s)	B12.FR.01, B12.FR.03	
System under test	Charging Station	
Description	This test case covers how the CSMS can remotely request the Charging Station to reset itself by sending a ResetRequest during a transaction. When ResetRequest "OnIdle" is send the charging stations schedules a reboot after all transactions are stopped. This could for example be necessary if the Charging Station is not functioning correctly.	
Purpose	To verify if the Charging Station is able to perform the reset mechanism while there is an ongoing transaction as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): **State** is *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **ResetResponse**	**1.** The OCTT sends a **ResetRequest** with **type** *OnIdle*
	3. Execute **Reusable State** *StopAuthorized*	
	4. Execute **Reusable State** *EVConnectedPostSession*	
	5. Execute **Reusable State** *EVDisconnected*	
	Notes(s): *Steps 4 and 5 will only be executed if TxStartPoint does not contain: EnergyTransferStarted, DataSigned, PowerPathClosed, or Authorized*	
	6. Execute **Reusable State** *ParkingBayUnoccupied*	
	Notes(s): *Step 6 will only be executed if TxStartPoint does not contain: EnergyTransferStarted, DataSigned, PowerPathClosed, Authorized, or EVConnected*	
	7. The Charging Station sends a **BootNotificationRequest**	**8.** The OCTT responds with a **BootNotificationResponse**
	9. The Charging Station notifies the CSMS about the current state of all connectors.	**10.** The OCTT responds accordingly.
	11. The Charging Station sends a **SecurityEventNotificationRequest**	**12.** The OCTT responds with a **SecurityEventNotificationResponse**

Test case name	**Reset Charging Station - With Ongoing Transaction - OnIdle**
Tool validations	* Step 2: Message **ResetResponse** - **status** *Scheduled* * Step 7: Message **BootNotificationRequest** - **reason** *ScheduledReset* * Step 9: Message: **StatusNotificationRequest** - If the transaction was stopped at step 3, then **connectorStatus** *Occupied* Else **connectorStatus** *Available* Message: **NotifyEventRequest** - If the transaction was stopped at step 3, then **eventData[0].actualValue** *"Occupied"* Else **eventData[0].actualValue** *"Available"* - **eventData[0].trigger** *Delta* - **eventData[0].component.name** *"Connector"* - **eventData[0].variable.name** *"AvailabilityState"* * Step 11: Message: **SecurityEventNotificationRequest** - **type** *StartupOfTheDevice* or *ResetOrReboot*
	Post scenario validations: - A message to report the state of a connector has been received for all connectors.

Table 43. Test Case Id: TC_B_22_CS

Test case name	**Reset Charging Station - With Ongoing Transaction - Immediate**	
Test case Id	TC_B_22_CS	
Use case Id(s)	B12	
Requirement(s)	B12.FR.02, B12.FR.04, E07.FR.03, B01.FR.03	
System under test	Charging Station	
Description	This test case covers how the CSMS can remotely request the Charging Station to reset itself by sending a ResetRequest during a transaction. When ResetRequest "Immediate" is send the charging stations will try to stop all transactions before rebooting. This could for example be necessary if the Charging Station is not functioning correctly.	
Purpose	To verify if the Charging Station is able to perform the reset mechanism as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): **State** is *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **ResetResponse**	**1.** The OCTT sends a **ResetRequest** with **type** *Immediate*
	3. The Charging Station sends a **TransactionEventRequest**	**4.** The OCTT responds with a **TransactionEventResponse**
	Note(s): - *Charging Station reboots*	
	5. The Charging Station sends a **BootNotificationRequest**	**6.** The OCTT responds with a **BootNotificationResponse** with **status** *Accepted*
	7. The Charging Station notifies the CSMS about the current state of all connectors.	**8.** The OCTT responds accordingly.

Test case name	**Reset Charging Station - With Ongoing Transaction - Immediate**
Tool validations	* Step 2: Message **ResetResponse** - **status** *Accepted* * Step 3: Message **TransactionEventRequest** - **eventType** *Ended* - **triggerReason** *ResetCommand* - **transactionInfo.chargingState** *EVConnected* - **transactionInfo.stoppedReason** *ImmediateReset* - **idToken** must be omitted * Step 5: Message **BootNotificationRequest** - **reason** *RemoteReset* * Step 7: For *<Configured connectorId>:* Message: **StatusNotificationRequest** - **connectorStatus** *Occupied* Message: **NotifyEventRequest** - **eventData[0].trigger** *Delta* - **eventData[0].actualValue** *"Occupied"* - **eventData[0].component.name** *"Connector"* - **eventData[0].variable.name** *"AvailabilityState"* For *<Other connector(s)>:* Message: **StatusNotificationRequest** - **connectorStatus** *Available* Message: **NotifyEventRequest** - **eventData[0].trigger** *Delta* - **eventData[0].actualValue** *"Available"* - **eventData[0].component.name** *"Connector"* - **eventData[0].variable.name** *"AvailabilityState"*
	Post scenario validations: - A message to report the state of a connector has been received for all connectors.

Table 44. Test Case Id: TC_B_23_CS

Test case name	**Reset Charging Station - Unavailable persists reset**	
Test case Id	TC_B_23_CS	
Use case Id(s)	B11	
Requirement(s)	B11.FR.01, B11.FR.02, B11.FR.03, B11.FR.04, B01.FR.03	
System under test	Charging Station	
Description	This test case covers how the CSMS can request the Charging Station to reset itself by sending a ResetRequest without any ongoing transaction and with the status of Inoperative. This could for example be necessary if the Charging Station is not functioning correctly.	
Purpose	To verify if the Charging Station is able to perform the reset mechanism as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: *Unavailable* for *<Configured connectorId>*	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **ResetResponse**	**1.** The OCTT sends a **ResetRequest** with **type** *OnIdle*
	Note(s): - *The Charging Station reboots*	
	3. The Charging Station sends a **BootNotificationRequest**	**4.** The OCTT responds with a **BootNotificationResponse** with **status** *Accepted*
	5. The Charging Station notifies the CSMS about the current state of all connectors.	**6.** The OCTT responds accordingly.

Test case name	**Reset Charging Station - Unavailable persists reset**
Tool validations	* Step 2: Message **ResetResponse** - **status** *Accepted* * Step 3: Message **BootNotificationRequest** **reason** *RemoteReset* * Step 5: For *<Configured connectorId>:* Message: **StatusNotificationRequest** - **connectorStatus** *Unavailable* Message: **NotifyEventRequest** - **eventData[0].trigger** *Delta* - **eventData[0].actualValue** *"Unavailable"* - **eventData[0].component.name** *"Connector"* - **eventData[0].variable.name** *"AvailabilityState"* For *<Other connector(s)>:* Message: **StatusNotificationRequest** - **connectorStatus** *Available* Message: **NotifyEventRequest** - **eventData[0].trigger** *Delta* - **eventData[0].actualValue** *"Available"* - **eventData[0].component.name** *"Connector"* - **eventData[0].variable.name** *"AvailabilityState"*
	Post scenario validations: - A message to report the state of a connector has been received for all connectors.

Table 45. Test Case Id: TC_B_24_CS

Test case name	**Reset Charging Station - Reserved persists reset**	
Test case Id	TC_B_24_CS	
Use case Id(s)	B11	
Requirement(s)	B11.FR.01, B11.FR.03, B11.FR.04, B11.FR.05, B01.FR.03	
System under test	Charging Station	
Description	This test case covers how the CSMS can request the Charging Station to reset itself by sending a ResetRequest without any ongoing transaction and with the status of Reserved. This could for example be necessary if the Charging Station is not functioning correctly.	
Purpose	To verify if the CSMS is able to perform the reset mechanism as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: Reserved for *<Configured connectorId>*	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **ResetResponse**	**1.** The OCTT sends a **ResetRequest** with **type** *OnIdle*
	Note(s): - *The Charging Station reboots*	
	3. The Charging Station sends a **BootNotificationRequest**	**4.** The OCTT responds with a **BootNotificationResponse** with **status** *Accepted*
	5. The Charging Station notifies the CSMS about the current state of all connectors.	**6.** The OCTT responds accordingly.
Tool validations	* Step 2: Message **ResetResponse** - **status** *Accepted* * Step 3: Message **BootNotificationRequest** **reason** *RemoteReset* * Step 5: For <*Configured connectorId>:* Message: **StatusNotificationRequest** - **connectorStatus** *Reserved* Message: **NotifyEventRequest** - **eventData[0].trigger** *Delta* - **eventData[0].actualValue** *"Reserved"* - **eventData[0].component.name** *"Connector"* - **eventData[0].variable.name** *"AvailabilityState"* For <*Other connector(s)>:* Message: **StatusNotificationRequest** - **connectorStatus** *Available* Message: **NotifyEventRequest** - **eventData[0].trigger** *Delta* - **eventData[0].actualValue** *"Available"* - **eventData[0].component.name** *"Connector"* - **eventData[0].variable.name** *"AvailabilityState"*	
	Post scenario validations: - A message to report the state of a connector has been received for all connectors.	

Table 46. Test Case Id: TC_B_41_CS

Test case name	**Reset Charging Station - With multiple ongoing transactions - OnIdle**	
Test case Id	TC_B_41_CS	
Use case Id(s)	B12	
Requirement(s)	B12.FR.01, B12.FR.03, E07.FR.03	
System under test	Charging Station	
Description	This test case covers how the CSMS can remotely request the Charging Station to reset itself by sending a ResetRequest during a transaction. When ResetRequest "OnIdle" is send the charging stations schedules a reboot after all transactions are stopped. This could for example be necessary if the Charging Station is not functioning correctly.	
Purpose	To verify if the Charging Station is able to perform the reset mechanism while there are multiple ongoing transactions as described at the OCPP specification.	
Prerequisite(s)	The Charging Station has more than one EVSE.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): **State** is *EnergyTransferStarted* for EVSE.id = 1 **State** is *EnergyTransferStarted* for EVSE.id = 2	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **ResetResponse**	**1.** The OCTT sends a **ResetRequest** with **type** *OnIdle*
	3. Execute **Reusable State** *StopAuthorized* for EVSE.id = 1	
	4. Execute **Reusable State** *EVConnectedPostSession* for EVSE.id = 1	
	5. Execute **Reusable State** *EVDisconnected* for EVSE.id = 1	
	6. Execute **Reusable State** *ParkingBayUnoccupied* for EVSE.id = 1	
	7. Execute **Reusable State** *StopAuthorized* for EVSE.id = 2	
	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.	
	9. Execute **Reusable State** *EVDisconnected* for EVSE.id = 2 Note(s): If TxStopPoint contains the value EVConnected. Then the transaction will have ended at the *EVDisconnected* state AND the Charging Station will proceed with resetting itself. Proceed to step 11 Else proceed with step 10	
	10. Execute **Reusable State** *ParkingBayUnoccupied* for EVSE.id = 2 Note(s): The transaction will end at this state, if it was not ended at an earlier state. Proceed to step 11.	
	11. The Charging Station sends a **BootNotificationRequest**	**12.** The OCTT responds with a **BootNotificationResponse**
	13. The Charging Station notifies the CSMS about the current state of all connectors.	**14.** The OCTT responds accordingly.

Test case name	**Reset Charging Station - With multiple ongoing transactions - OnIdle**
Tool validations	* Step 2: Message **ResetResponse** - **status** *Scheduled* * Step 11: Message **BootNotificationRequest** - **reason** *ScheduledReset* * Step 13: Message: **StatusNotificationRequest** - If the transaction was stopped at step 3, then **connectorStatus** *Occupied* Else **connectorStatus** *Available* Message: **NotifyEventRequest** - If the transaction was stopped at step 3, then **eventData[0].actualValue** *"Occupied"* Else **eventData[0].actualValue** *"Available"* - **eventData[0].trigger** *Delta* - **eventData[0].component.name** *"Connector"* - **eventData[0].variable.name** *"AvailabilityState"*
	Post scenario validations: - A message to report the state of a connector has been received for all connectors.

Table 47. Test Case Id: TC_B_25_CS

Test case name	**Reset EVSE - Without ongoing transaction**	
Test case Id	TC_B_25_CS	
Use case Id(s)	B11	
Requirement(s)	B11.FR.01, B11.FR.08, B11.FR.10	
System under test	Charging Station	
Description	This test case covers how the CSMS can request the Charging Station to reset an EVSE by sending a ResetRequest without any ongoing transaction. This could for example be necessary if the Charging Station is not functioning correctly.	
Purpose	To verify if the Charging Station is able to perform the reset mechanism as described at the OCPP specification.	
Prerequisite(s)	Individual resetting EVSE supported	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **ResetResponse**	**1.** The OCTT sends a **ResetRequest** with **type** *OnIdle* and *evseId*<*Configured evseId*>
	Note(s): - *<Configured evseId> reboots*	
Tool validations	* Step 2: Message **ResetResponse** - **status** *Accepted*	
	Post scenario validations: - N/a	

Table 48. Test Case Id: TC_B_26_CS

Test case name	**Reset EVSE - With Ongoing Transaction - OnIdle**	
Test case Id	TC_B_26_CS	
Use case Id(s)	B12	
Requirement(s)	B12.FR.01, B12.FR.07	
System under test	Charging Station	
Description	This test case covers how the CSMS can remotely request the Charging Station to reset an EVSE by sending a ResetRequest during a transaction. When ResetRequest "OnIdle" is send the charging stations schedules a reboot after all transactions are stopped. This could for example be necessary if the Charging Station is not functioning correctly.	
Purpose	To verify if the Charging Station is able to perform the reset mechanism as described at the OCPP specification.	
Prerequisite(s)	Individual resetting EVSE supported	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): **State** is *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **ResetResponse**	**1.** The OCTT sends a **ResetRequest** with **type** *OnIdle* and **evseId** *<Configured evseId>*
	3. Execute **Reusable State** *StopAuthorized*	
	4. Execute **Reusable State** *EVConnectedPostSession*	
	5. Execute **Reusable State** *EVDisconnected*	
	6. Execute **Reusable State** *ParkingBayUnoccupied*	
Tool validations	* Step 2: Message **ResetResponse** - **status** *Scheduled*	
	Post scenario validations: N/a	

Table 49. Test Case Id: TC_B_27_CS

Test case name	**Reset EVSE - With Ongoing Transaction - Immediate**	
Test case Id	TC_B_27_CS	
Use case Id(s)	B12	
Requirement(s)	B12.FR.02, B12.FR.08, E07.FR.03	
System under test	Charging Station	
Description	This test case covers how the CSMS can remotely request the Charging Station to reset an EVSE by sending a ResetRequest during a transaction. When ResetRequest "Immediate" is send the charging stations will try to stop all transactions before rebooting. This could for example be necessary if the Charging Station is not functioning correctly.	
Purpose	To verify if the Charging Station is able to perform the reset mechanism as described at the OCPP specification.	
Prerequisite(s)	Individual resetting EVSE supported	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): **State** is *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **ResetResponse**	**1.** The OCTT sends a **ResetRequest** with **type** *Immediate* and *evseId*<*Configured evseId*>
	3. The Charging Station sends a **TransactionEventRequest**	**4.** The OCTT responds with a **TransactionEventResponse**
	Note(s): - *The EVSE reboots*	
Tool validations	* Step 2: Message **ResetResponse** - **status** *Accepted* * Step 3: Message **TransactionEventRequest** - **eventType** *Ended* - **triggerReason** *ResetCommand* - **transactionInfo.chargingState** *EVConnected* - **transactionInfo.stoppedReason** *ImmediateReset*	
	Post scenario validations: - N/a	

Table 50. Test Case Id: TC_B_28_CS

Test case name	**Reset EVSE - Not Supported**	
Test case Id	TC_B_28_CS	
Use case Id(s)	B11, B12	
Requirement(s)	B11.FR.01, B11.FR.09, B12.FR.01, B12.FR.09	
System under test	Charging Station	
Description	This test case covers how the CSMS can request the Charging Station to reset an EVSE by sending a ResetRequest while it is not supported by the Charging Station.	
Purpose	To verify if the Charging Station is able to perform the reset mechanism as described at the OCPP specification.	
Prerequisite(s)	Charging Station does not support resetting individual EVSE	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **ResetResponse**	**1.** The OCTT sends a **ResetRequest** with **type** *OnIdle* and *evseId*<*Configured evseId*>
Tool validations	* Step 2: Message **ResetResponse** - **status** *Rejected*	
	Post scenario validations: - N/a	

Table 51. Test Case Id: TC_B_29_CS

Test case name	**Reset EVSE - With ongoing transaction - Not Supported**	
Test case Id	TC_B_29_CS	
Use case Id(s)	B11	
Requirement(s)	B12.FR.01, B12.FR.09	
System under test	Charging Station	
Description	This test case covers how the CSMS can request the Charging Station to reset an EVSE by sending a ResetRequest with ongoing transaction while it is not supported by the Charging Station.	
Purpose	To verify if the Charging Station is able to perform the reset mechanism as described at the OCPP specification.	
Prerequisite(s)	Charging Station does not support resetting individual EVSE	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): **State is** *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **ResetResponse**	**1.** The OCTT sends a **ResetRequest** with **type** *OnIdle* and **evseId** *<Configured evseId>*
Tool validations	* Step 2: Message **ResetResponse** - **status** *Rejected*	
	Post scenario validations: - N/a	

Table 52. Test Case Id: TC_B_43_CS

Test case name	**Set new NetworkConnectionProfile - Rejected**	
Test case Id	TC_B_43_CS	
Use case Id(s)	B09	
Requirement(s)	B09.FR.02	
System under test	Charging Station	
Description	The CSMS updates the connection details on the Charging Station. For instance in preparation of a migration to a new CSMS.	
Purpose	To verify if the Charging Station is able to reject when the CSMS tries to set a network connection profile containing invalid data.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SetNetworkProfileResponse**	**1.** The OCTT sends a **SetNetworkProfileRequest** with: - **configurationSlot** is *999* - **connectionData.messageTimeout** *<Configured messageTimeout>* - **connectionData.ocppCsmsUrl** *<Configured ocppCsmsUrl>* - **connectionData.ocppInterface** *<Configured ocppInterface>* - **connectionData.ocppVersion** *OCPP20* - **connectionData.securityProfile** *<Configured securityProfile>*
Tool validations	* Step 2: Message **SetNetworkProfileResponse** - **status** *Rejected*	
	Post scenario validations: - N/a	

Table 53. Test Case Id: TC_B_45_CS

Test case name	**Migrate to new ConnectionProfile - Success - Same CSMS Root**	
Test case Id	TC_B_45_CS	
Use case Id(s)	B09, B10	
Requirement(s)	B09.FR.01,B10.FR.01,B10.FR.04,B10.FR.06	
System under test	Charging Station	
Description	The CSMS updates the connection details on the Charging Station. For instance in preparation of a migration to a new CSMS.	
Purpose	To verify if the Charging Station is able to migrate to another network connection profile slot.	
Prerequisite(s)	At least two configuration slots for networkConnectionProfiles must be supported	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SetNetworkProfileResponse**	**1.** The OCTT sends a **SetNetworkProfileRequest** with **configurationSlot** is *<Configured configurationSlot>* or *<Configured configurationSlot2>* depending on which one is already in use - **connectionData.messageTimeout** *<Configured messageTimeout>* or *<Configured messageTimeout2>* - **connectionData.ocppCsmsUrl** *<ocppCsmsUrl that is not currently active>* - **connectionData.ocppInterface** *<Configured ocppInterface>* or *<Configured ocppInterface2>* - **connectionData.ocppVersion** *OCPP20* - **connectionData.securityProfile** *<Configured securityProfile>* or *<Configured securityProfile2>*
	4. The Charging Station responds with a **SetVariablesResponse**	**3.** The OCTT sends a **SetVariablesRequest** with **variable.name** is *"NetworkConfigurationPriority"* **component.name** is *"OCPPCommCtrlr"* **attributeValue** is *Configured slot from Step 1, the previously configured slot*
	6. The Charging Station responds with a **ResetResponse**	**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.*
	7. Execute **Reusable State** *Booted* Note(s): *- The Charging Station connects using the <Configured connectionData2>.*	

Test case name	**Migrate to new ConnectionProfile - Success - Same CSMS Root**
Tool validations	* Step 2: Message **SetNetworkProfileResponse** - **status** *Accepted* * Step 4: Message **SetVariablesResponse** - **setVariableResult[0].attributeStatus** *Accepted* OR *RebootRequired* * Step 6: Message **ResetResponse** - **status** *Accepted*
	Post scenario validations: - N/a

Table 54. Test Case Id: TC_B_46_CS

Test case name	**Migrate to new ConnectionProfile - Fallback mechanism - Same CSMS Root**	
Test case Id	TC_B_46_CS	
Use case Id(s)	B10	
Requirement(s)	B10.FR.03,B10.FR.04	
System under test	Charging Station	
Description	The CSMS updates the connection details on the Charging Station. For instance in preparation of a migration to a new CSMS.	
Purpose	To verify if the Charging Station is able to use the fallback mechanism when it is unable to connect with the first network connection profile slot.	
Prerequisite(s)	At least two configuration slots for networkConnectionProfiles must be supported	
Before (Preparations)	**Configuration State:** **OCPPCommCtrlr.NetworkProfileConnectionAttempts** is *2*	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SetNetworkProfileResponse**	**1.** The OCTT sends a **SetNetworkProfileRequest** with **configurationSlot** is *<Configured configurationSlot>* or *<Configured configurationSlot2>* depending on which one is already in use - **connectionData.messageTimeout** *<Configured messageTimeout>* or *<Configured messageTimeout2>* - **connectionData.ocppCsmsUrl** *<ocppCsmsUrl that is not currently active>* - **connectionData.ocppInterface** *<Configured ocppInterface>* or *<Configured ocppInterface2>* - **connectionData.ocppVersion** *OCPP20* - **connectionData.securityProfile** *<Configured securityProfile>* or *<Configured securityProfile2>*
	4. The Charging Station responds with a **SetVariablesResponse**	**3.** The OCTT sends a **SetVariablesRequest** with **variable.name** is *"NetworkConfigurationPriority"* **component.name** is *"OCPPCommCtrlr"* **attributeValue** is *Configured slot from Step 1, the previously configured slot*
	6. The Charging Station responds with a **ResetResponse**	**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.*

Test case name	**Migrate to new ConnectionProfile - Fallback mechanism - Same CSMS Root**	
		7. The OCTT will NOT respond to the two connection request from the Charging Station from the first connectionSlot.
		8. The OCTT will accept the connection request from the Charging Station from the second connectionSlot.
	Note(s): *Set the <Configured Long Operation Time Out> so that Steps 7 and 8 can be completed in this time period.*	
	9. Execute **Reusable State** *Booted* Note(s): *- The Charging Station connects using the <Configured connectionData>.*	
Tool validations	* Step 2: Message **SetNetworkProfileResponse** - **status** *Accepted* * Step 4: Message **SetVariablesResponse** - **setVariableResult[0].attributeStatus** *Accepted* OR *RebootRequired* * Step 6: Message **ResetResponse** - **status** *Accepted*	
	Post scenario validations: - N/a	

Table 55. Test Case Id: TC_B_47_CS

Test case name	**Migrate to new ConnectionProfile - Fallback after NetworkProfileConnectionAttempts per NetworkConfigurationPriority failed - New CSMS Root - New CSMS**
Test case Id	TC_B_47_CS
Use case Id(s)	B09,B10,M05
Requirement(s)	B10.FR.07,M05.FR.15,M05.FR.16
System under test	Charging Station
Description	The CSMS updates the connection details on the Charging Station. For instance in preparation of a migration to a new CSMS.
Purpose	To verify if the Charging Station is able to correctly handle the fallback mechanism in the case it fails to connect to the other CSMS.
Prerequisite(s)	- The Charging Station supports AS-2: AdditionalRootCertificateCheck. - Configured (new) CSMS Root certificate 2 must be signed by the configured (old) CSMS Root certificate 2. - At least two configuration slots for networkConnectionProfiles must be supported
Before (Preparations)	**Configuration State:** **OCPPCommCtrlr.NetworkProfileConnectionAttempts** is *1*
	Memory State: *CertificateInstalled* for certificateType *CSMSRootCertificate* and certificate *<Configured (new) CSMS Root certificate 2>*
	Reusable State(s): N/a

Test case name	**Migrate to new ConnectionProfile - Fallback after NetworkProfileConnectionAttempts per NetworkConfigurationPriority failed - New CSMS Root - New CSMS**	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SetNetworkProfileResponse**	**1.** The OCTT sends a **SetNetworkProfileRequest** with **configurationSlot** is *<Configured configurationSlot>* or *<Configured configurationSlot2>* depending on which one is already in use - **connectionData.messageTimeout** *<Configured messageTimeout>* or *<Configured messageTimeout2>* - **connectionData.ocppCsmsUrl** *<ocppCsmsUrl that is not currently active>* - **connectionData.ocppInterface** *<Configured ocppInterface>* or *<Configured ocppInterface2>* - **connectionData.ocppVersion** *OCPP20* - **connectionData.securityProfile** *<Configured securityProfile>* or *<Configured securityProfile2>*
	4. The Charging Station responds with a **SetVariablesResponse**	**3.** The OCTT sends a **SetVariablesRequest** with **variable.name** is *"NetworkConfigurationPriority"* **component.name** is *"OCPPCommCtrlr"* **attributeValue** is *Configured slot from Step 1, the previously configured slot*
	6. The Charging Station responds with a **ResetResponse**	**5.** The OCTT sends a **ResetRequest** with **type** *OnIdle*
	8. During the TLS handshake the Charging Station validates the CSMS certificate. Note(s): *- This connection attempt fails, because the Charging Station will use the new CSMS Root certificate to validate the CSMS certificate.*	**7.** During the TLS handshake the OCTT provides a CSMS certificate which is signed by the *<Configured old CSMS Root certificate>*
	9. The Charging Station switches back to the previous networkprofile configuration and validates the CSMS certificate, using the (fallback) CSMS Root certificate. Note(s): *- This connection attempt succeeds, because the Charging Station will now use the (old) CSMS Root certificate to validate the CSMS certificate.*	
	10. Execute **Reusable State** *Booted*	
	12. The Charging Station responds with a **GetInstalledCertificateIdsResponse**	**11.** The OCTT sends a **GetInstalledCertificateIdsRequest** with **certificateType** is *CSMSRootCertificate*

Test case name	**Migrate to new ConnectionProfile - Fallback after NetworkProfileConnectionAttempts per NetworkConfigurationPriority failed - New CSMS Root - New CSMS**
Tool validations	* Step 6: Message **ResetResponse** - **status** *Accepted* * Step 12: Message: **GetInstalledCertificateIdsResponse** - **status** must be *Accepted* - **certificateHashDataChain** must contain an entry with following values: - **certificateType** is *CSMSRootCertificate* - **certificateHashData** contains *<HashData from configured old CSMS Root certificate>*
	Post scenario validations: - N/a

Table 56. Test Case Id: TC_B_49_CS

Test case name	**Migrate to new ConnectionProfile - Fallback after NetworkProfileConnectionAttempts per NetworkConfigurationPriority failed - Same CSMS Root**
Test case Id	TC_B_49_CS
Use case Id(s)	B10
Requirement(s)	B10.FR.07
System under test	Charging Station
Description	The CSMS updates the connection details on the Charging Station. For instance in preparation of a migration to a new CSMS.
Purpose	To verify if the Charging Station is able to correctly handle the fallback mechanism in the case it fails to connect to the other CSMS.
Prerequisite(s)	- The Charging Station supports C-47: mechanism implemented & Reconnect after NetworkProfileConnectionAttempts - At least two configuration slots for networkConnectionProfiles must be supported
Before (Preparations)	**Configuration State:** **OCPPCommCtrlr.NetworkProfileConnectionAttempts** is *1* **OCPPCommCtrlr.RetryBackOffRepeatTimes** is *0* **OCPPCommCtrlr.RetryBackOffRandomRange** is *0* **OCPPCommCtrlr.RetryBackOffWaitMinimum** is *<Configured RetryBackOffWaitMinimum>*
	Memory State: N/a
	Reusable State(s): N/a

Test case name	**Migrate to new ConnectionProfile - Fallback after NetworkProfileConnectionAttempts per NetworkConfigurationPriority failed - Same CSMS Root**	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SetNetworkProfileResponse**	**1.** The OCTT sends a **SetNetworkProfileRequest** with **configurationSlot** is *<Configured configurationSlot>* or *<Configured configurationSlot2>* depending on which one is already in use - **connectionData.messageTimeout** *<Configured messageTimeout>* or *<Configured messageTimeout2>* - **connectionData.ocppCsmsUrl** *<ocppCsmsUrl that is not currently active>* - **connectionData.ocppInterface** *<Configured ocppInterface>* or *<Configured ocppInterface2>* - **connectionData.ocppVersion** *OCPP20* - **connectionData.securityProfile** *<Configured securityProfile>* or *<Configured securityProfile2>*
	4. The Charging Station responds with a **SetVariablesResponse**	**3.** The OCTT sends a **SetVariablesRequest** with **variable.name** is *"NetworkConfigurationPriority"* **component.name** is *"OCPPCommCtrlr"* **attributeValue** is *Configured slot from Step 1, the previously configured slot*
	6. The Charging Station responds with a **ResetResponse**	**5.** The OCTT sends a **ResetRequest** with **type** *OnIdle*
	7. The Charging Station tries to connect to the alternative internal OCTT endpoint. Note(s): - *Make sure to set the <Configured Long Operation Time Out> to be the time required for the CS to revert to the previous network profile configuration.*	**8.** The connection attempt is not accepted by the OCTT.
	9. The Charging Station switches back to the previous networkprofile configuration and reconnects to the OCTT.	**10.** The connection attempt is not accepted by the OCTT.
	11. The Charging Station waits for the duration of the configured RetryBackOffWaitMinimum and reconnects to the OCTT.	**12.** The connection attempt is accepted by the OCTT.
	13. Execute **Reusable State** *Booted*	
Tool validations	* Step 6: Message **ResetResponse** - **status** *Accepted*	
	Post scenario validations: - N/a	

Table 57. Test Case Id: TC_B_50_CS

Test case name	**Migrate to new ConnectionProfile - Success - New CSMS Root - New CSMS**	
Test case Id	TC_B_50_CS	
Use case Id(s)	B10,M05	
Requirement(s)	M05.FR.13	
System under test	Charging Station	
Description	The CSMS updates the connection details on the Charging Station. For instance in preparation of a migration to a new CSMS.	
Purpose	To verify if the Charging Station is able to correctly handle migrating to the new CSMS using a new CSMS Root certificate to validate the server certificate.	
Prerequisite(s)	- At least two configuration slots for networkConnectionProfiles must be supported AND - The Charging Station must be connected using either security profile 2 or 3.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: CertificateInstalled for certificateType *CSMSRootCertificate* and certificate *<Configured (new) CSMS Root certificate 2>*	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SetNetworkProfileResponse**	**1.** The OCTT sends a **SetNetworkProfileRequest** with **configurationSlot** is *<Configured configurationSlot>* or *<Configured configurationSlot2>* 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 securityProfile>* or *<Configured securityProfile2>*
	4. The Charging Station responds with a **SetVariablesResponse**	**3.** The OCTT sends a **SetVariablesRequest** with **variable.name** is *"NetworkConfigurationPriority"* **component.name** is *"OCPPCommCtrlr"* **attributeValue** is *Configured slot from Step 1, the previously configured slot*
	6. The Charging Station responds with a **ResetResponse**	**5.** The OCTT sends a **ResetRequest** with **type** *OnIdle*
	7. The Charging Station connects to the configured alternative internal OCTT endpoint. Note(s): *- During the TLS handshake the Charging Station validates and accepts the CSMS certificate, signed by the <Configured (new) CSMS Root certificate 2>.*	**8.** The connection attempt is accepted by the OCTT.
	9. Execute **Reusable State** *Booted*	
Tool validations	* Step 6: Message **ResetResponse** - **status** *Accepted*	
	Post scenario validations: - N/a	

Table 58. Test Case Id: TC_B_51_CS

Test case name	**Status change during offline period - > Offline Threshold**	
Test case Id	TC_B_51_CS	
Use case Id(s)	B04	
Requirement(s)	B04.FR.01	
System under test	Charging Station	
Description	A Charging Station sends a notification to the CSMS to inform the CSMS about a Connector status change. This can be done in two ways. Via a StatusNotificationRequest or a NotifyEventRequest from the device model. When the Charging Station is connected again to the CSMS after having been offline, and the Charging Station was longer offline than the configured threshold, it will report the status of every connector.	
Purpose	To verify whether the Charging Station reports the status of all connectors after having been offline for longer than the configured threshold with the configuration variable **OfflineThreshold**.	
Prerequisite(s)	If the Charging Station does not have more than one EVSE, this testcase will be equal to TC_B_52_CS.	
Before (Preparations)	**Configuration State:** **OCPPCommCtrlr.OfflineThreshold** is *<Configured offlineThreshold>* **OCPPCommCtrlr.RetryBackOffWaitMinimum** is *<Configured offlineThreshold> + 2 seconds* **OCPPCommCtrlr.RetryBackOffRandomRange** is *0*	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
		1. *The OCTT closes the WebSocket connection AND does not accept a reconnect.*
	2. Manual Action: *Connect the EV and EVSE.*	
		3. *The OCTT accepts reconnection attempt from the Charging Station, after the configured threshold has been exceeded.*
	4. The Charging Station notifies the CSMS about the current state of all connectors.	**5.** The OCTT responds accordingly.

Test case name	**Status change during offline period - > Offline Threshold**
Tool validations	* Step 4: *Configured EVSE/Connector:* Message: **StatusNotificationRequest** - **connectorStatus** *Occupied* - **evseId** *<Configured evseId>* - **connectorId** *<Configured connectorId>* Message: **NotifyEventRequest** - **eventData[0].trigger** *Delta* - **eventData[0].actualValue** *"Occupied"* - **eventData[0].component.name** *"Connector"* - **eventData[0].variable.name** *"AvailabilityState"* - **eventData[0].evse.id** *<Configured evseId>* - **eventData[0].connectorId** *<Configured connectorId>* *All other EVSE/Connector(s):* Message: **StatusNotificationRequest** - **connectorStatus** *Available* Message: **NotifyEventRequest** - **eventData[0].trigger** *Delta* - **eventData[0].actualValue** *"Available"* - **eventData[0].component.name** *"Connector"* - **eventData[0].variable.name** *"AvailabilityState"*
	Post scenario validations: N/a

Table 59. Test Case Id: TC_B_52_CS

Test case name	**Status change during offline period - < Offline Threshold**	
Test case Id	TC_B_52_CS	
Use case Id(s)	B04	
Requirement(s)	B04.FR.02	
System under test	Charging Station	
Description	A Charging Station sends a notification to the CSMS to inform the CSMS about a Connector status change. This can be done in two ways. Via a StatusNotificationRequest or a NotifyEventRequest from the device model. When the Charging Station is connected again to the CSMS after having been offline, and the Charging Station was shorter offline than the configured threshold, it will report the status of all connector that received a status change.	
Purpose	To verify whether the Charging Station reports the status of connectors that received a status change after having been offline for shorter than the configured threshold with the configuration variable **OfflineThreshold**.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** **OCPPCommCtrlr.OfflineThreshold** is *<Configured offlineThreshold>* **OCPPCommCtrlr.RetryBackOffWaitMinimum** is *<Configured offlineThreshold> - 2 seconds* **OCPPCommCtrlr.RetryBackOffRandomRange** is *0*	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. *The OCTT closes the WebSocket connection AND does not accept a reconnect.*	
	2. Manual Action: *Connect the EV and EVSE.*	
	3. *The OCTT accepts reconnection attempt from the Charging Station.*	
	4. The Charging Station notifies the CSMS about the current state of the configured connector.	**5.** The OCTT responds accordingly.
Tool validations	* Step 3: Message: **StatusNotificationRequest** - **connectorStatus** *Occupied* - **evseId** *<Configured evseId>* - **connectorId** *<Configured connectorId>* Message: **NotifyEventRequest** - **eventData[0].trigger** *Delta* - **eventData[0].actualValue** *"Occupied"* - **eventData[0].component.name** *"Connector"* - **eventData[0].variable.name** *"AvailabilityState"* - **eventData[0].evse.id** *<Configured evseId>* - **eventData[0].connectorId** *<Configured connectorId>*	
	Post scenario validations: N/a	

Table 60. Test Case Id: TC_B_53_CS

Test case name	**Get Base Report - Test mandatory DM variables via FullInventory**	
Test case Id	TC_B_53_CS	
Use case Id(s)	B07	
Requirement(s)	Chapter Referenced Components and Variables	
System under test	Charging Station	
Description	CSMS requests a FullInventory base report.	
Purpose	To test that Charging Station supports all required DM variables.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. CS responds with: **GetBaseReportResponse** with **status** = *Accepted* **3.** CS sends one or more **NotifyReportRequest** messages to report all its component/variables.	**1.** OCTT requests a **GetBaseReportRequest** with: **reportBase** = *FullInventory* and **requestId** = <Generated requestId> **4.** OCTT responds with a **NotifyReportResponse** for each **NotifyReportRequest**
Tool validations	* Step 2: Message: **GetBaseReportResponse** with: - **status** = *Accepted* - **statusInfo** is absent or **statusInfo** = *"NoError"*	
	* step 3: Message: **NotifyReportRequest** with: - **requestId** = <Generated requestId> - **generatedAt** = <time of generation at charging station> - **seqNo** = *0*	
	While **tbc** = *true*	Message: **NotifyReportRequest** - **seqNo** is incremented by one
	Post scenario validations: The OCTT checks that: - the components / variables that are required according to the OCPP specification are implemented; - for each component/variable, where **variableCharacteristics.dataType** is set to **OptionList**, **SequenceList** or **MemberList**, the **variableCharacteristics.valuesList** is not omitted or empty; - for each component/variable, where **variableCharacteristics.dataType** is **OptionList**, **SequenceList** or **MemberList**, the **variableAttribute.value** is allowed based on the values in the provided **variableCharacteristics.valuesList**; - for variables with **mutability** set to *WriteOnly* the variableAttribute.value is omitted in the **NotifyReportRequest**.	

Table 61. Test Case Id: TC_B_54_CS

Test case name	**Get Custom Report - with component/variable, but no instance**	
Test case Id	TC_B_54_CS	
Use case Id(s)	B08	
Requirement(s)	B08.FR.01, B08.FR.03, B08.FR.04, B08.FR.11, B08.FR.12, B08.FR.14	
System under test	Charging Station	
Description	CSMS requests a custom report for ItemsPerMessage of DeviceDataCtrlr.	
Purpose	To test that Charging Station will send all instances if instance is not given.	
Prerequisite(s)	Charging Station has implemented custom reporting (use case B08).	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. Charging Station responds: **GetReportResponse**	**1.** OCTT sends **GetReportRequest** with: - **requestId** = *<Generated requestId>* - **componentVariable.component.name**= *"DeviceDataCtrlr"* - **componentVariable.variable.name** = *"ItemsPerMessage"*
	3. Charging Station sends **NotifyReportRequest**	**4.** OCTT responds with **NotifyReportResponse**
Tool validations	* Step 2: Message: **GetReportResponse** with: - **status** = *Accepted* - **statusInfo** is absent or **statusInfo** = *"NoError"*	
	* step 3: Message: **NotifyReportRequest** with: - **reportData[0].component.name** = *"DeviceDataCtrlr"* - **reportData[0].variable.name** = *"ItemsPerMessage"* - **reportData[0].variable.instance** = *"GetReport"* - **reportData[1].component.name** = *"DeviceDataCtrlr"* - **reportData[1].variable.name** = *"ItemsPerMessage"* - **reportData[1].variable.instance** = *"GetVariable"* *Note: for AvailabilityState there will only be an _Actual* value. It does not make any sense to have a *MinSet*, *MaxSet* or *Target* value for it._	
	Post scenario validations: N/A	

Table 62. Test Case Id: TC_B_55_CS

Test case name	**Get Custom Report - with component/variable/instance**	
Test case Id	TC_B_55_CS	
Use case Id(s)	B08	
Requirement(s)	B08.FR.01, B08.FR.03, B08.FR.04, B08.FR.11, B08.FR.12, B08.FR.14	
System under test	Charging Station	
Description	CSMS requests a custom report for ItemsPerMessage of DeviceDataCtrlr.	
Purpose	To test that Charging Station will send one instances if instance is given.	
Prerequisite(s)	Charging Station has implemented custom reporting (use case B08).	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. Charging Station responds: **GetReportResponse**	**1.** OCTT sends **GetReportRequest** with: - **requestId** = *<Generated requestId>* - **componentVariable.component.name**= *"DeviceDataCtrlr"* - **componentVariable.variable.name** = *"ItemsPerMessage"* - **componentVariable.instance** = *"GetReport"*
	3. Charging Station sends **NotifyReportRequest**	**4.** OCTT responds with **NotifyReportResponse**
Tool validations	* Step 2: Message: **GetReportResponse** with: - **status** = *Accepted* - **statusInfo** is absent or **statusInfo** = *"NoError"*	
	* step 3: Message: **NotifyReportRequest** with: - **reportData[0].component.name** = *"DeviceDataCtrlr"* - **reportData[0].variable.name** = *"ItemsPerMessage"* - **reportData[0].variable.instance** = *"GetReport"* *Note: for AvailabilityState there will only be an _Actual* value. It does not make any sense to have a *MinSet*, *MaxSet* or *Target* value for it._	
	Post scenario validations: N/A	

Table 63. Test Case Id: TC_B_56_CS

Test case name	**Get Custom Report - with component/variable, but no evseId**	
Test case Id	TC_B_56_CS	
Use case Id(s)	B08	
Requirement(s)	B08.FR.01, B08.FR.03, B08.FR.04, B08.FR.11, B08.FR.12, B08.FR.14	
System under test	Charging Station	
Description	CSMS requests a custom report for AvailabilityState of EVSE	
Purpose	To test that Charging Station will send all EVSEs when evseId is not given.	
Prerequisite(s)	Charging Station has implemented custom reporting (use case B08).	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. Charging Station responds: **GetReportResponse**	**1.** OCTT sends **GetReportRequest** with: - **requestId** = *<Generated requestId>* - **componentVariable.component.name**= *"EVSE"* - **componentVariable.variable.name** = *"AvailabilityState"*
	3. Charging Station sends **NotifyReportRequest**	**4.** OCTT responds with **NotifyReportResponse**
Tool validations	* Step 2: Message: **GetReportResponse** with: - **status** = *Accepted* - **statusInfo** is absent or **statusInfo** = *"NoError"*	
	* step 3: Message: **NotifyReportRequest** with: - **reportData[i].component.name** = *"EVSE"* - **reportData[i].variable.name** = *"AvailabilityState"* - **number of EVSEs** = *<Configured EVSE count>* *Note: for AvailabilityState there will only be an _Actual* value. It does not make any sense to have a *MinSet*, *MaxSet* or *Target* value for it._	
	Post scenario validations: N/A	

Table 64. Test Case Id: TC_B_57_CS

Test case name	**Network Reconnection - After connection loss**	
Test case Id	TC_B_57_CS	
Use case Id(s)	Part 4 section 5.3. Reconnecting	
Requirement(s)	Described at section 5.3.	
System under test	Charging Station	
Description	When the connection is lost, the Charging Station SHALL try to reconnect. When reconnecting, the Charging Station SHALL use an increasing back-off time until it has successfully reconnected.	
Purpose	To verify if the Charging Station is able to reconnect to the CSMS using the described OCPP reconnecting mechanism from part 4.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** **OCPPCommCtrlr.RetryBackOffRepeatTimes** is *2* **OCPPCommCtrlr.RetryBackOffRandomRange** is *0* **OCPPCommCtrlr.RetryBackOffWaitMinimum** is *<Configured RetryBackOffWaitMinimum>*	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The OCTT closes the websocket connection.	
	2. The Charging Station waits for the duration of the configured RetryBackOffWaitMinimum and reconnects to the OCTT.	**3.** The connection attempt is accepted by the OCTT.
	4. The OCTT closes the websocket connection.	
	5. The Charging Station waits for the duration of the configured RetryBackOffWaitMinimum and reconnects to the OCTT.	**6.** The connection attempt is not accepted by the OCTT.
	7. The Charging Station waits for the duration of the at step 5 doubled RetryBackOffWaitMinimum and reconnects to the OCTT.	**8.** The connection attempt is accepted by the OCTT.
Tool validations	* Step 2: - The reconnection time is at least the configured RetryBackOffWaitMinimum. * Step 7: - The reconnection time is at least 2 times the reconnection time from step 5.	
	Post scenario validations: - N/a	

2.4. C Authorization

Table 65. Test Case Id: TC_C_02_CS

Test case name	**Local start transaction - Authorization Invalid/Unknown**	
Test case Id	TC_C_02_CS	
Use case Id(s)	C01 OR C04 OR C06	
Requirement(s)	C01.FR.02 OR C06.FR.02	
System under test	Charging Station	
Description	When a Charging Station needs to charge an EV, it needs to authorize the EV Driver first at the CSMS before the charging can be started or stopped.	
Purpose	To verify whether the Charging Station is able to handle receiving an invalid idToken.	
Prerequisite(s)	- The Charging Station supports at least one authorization method described at the following Use cases; C01, C04, C06. - The Charging Station does NOT have a cable lock, which prevents the EV driver to connect the EV and EVSE before authorization.	
Before (Preparations)	**Configuration State:** **AuthCtrlr.AuthEnabled** is *true* (If implemented AND ReadWrite) **AuthCtrlr.DisableRemoteAuthorization** is *false* (If implemented)	
	Memory State: N/a	
	Reusable State(s): **State is** *EVConnectedPreSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Present idToken.*	
	1. The Charging Station sends an **AuthorizeRequest**	**2.** The OCTT responds with an **AuthorizeResponse** with **idTokenInfo.status** *Invalid*
	Note(s): *- The Charging Station SHALL NOT send a TransactionEventRequest message after the AuthorizeRequest from step 1.* *- The OCTT waits <Configured message timeout> seconds, before ending the testcase.*	
Tool validations	* Step 1: Message: **AuthorizeRequest** - **idToken.idToken** *<Configured invalid_idtoken_idtoken>* - **idToken.type** *<Configured invalid_idtoken_type>*	
	Post scenario validations: N/a	

Table 66. Test Case Id: TC_C_05_CS

Test case name	**Local start transaction - Authorization invalid - Cable lock**	
Test case Id	TC_C_05_CS	
Use case Id(s)	C01 OR C04 OR C06	
Requirement(s)	C01.FR.02 OR C06.FR.02	
System under test	Charging Station	
Description	When a Charging Station needs to charge an EV, it needs to authorize the EV Driver first before the charging can be started or stopped.	
Purpose	To verify whether a Charging Station with a cable lock, which prevents the EV driver to connect the EV and EVSE before authorization, is able to handle receiving an invalid idToken.	
Prerequisite(s)	- The Charging Station has a cable lock, which prevents the EV driver to connect the EV and EVSE before authorization. - The Charging Station supports at least one authorization method described at the following Use cases; C01, C04, C06. - The Charging Station does NOT have the following configuration; TxStartPoint ReadOnly AND value Authorized is NOT set.	
Before (Preparations)	**Configuration State:** **AuthCtrlr.AuthEnabled** is *true* (If implemented AND ReadWrite) **AuthCtrlr.DisableRemoteAuthorization** is *false* (If implemented)	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Present idToken.*	
	1. The Charging Station sends an **AuthorizeRequest**	**2.** The OCTT responds with an **AuthorizeResponse** with **idTokenInfo.status** *Invalid*
	Note(s): *- The Charging Station SHALL NOT send a TransactionEventRequest message after the AuthorizeRequest from step 1.* *- The OCTT waits <Configured message timeout> seconds, before ending the testcase.*	
Tool validations	* Step 1: Message: **AuthorizeRequest** - **idToken.idToken** *<Configured invalid_idtoken_idtoken>* - **idToken.type** *<Configured invalid_idtoken_type>*	
	Post scenario validations: N/a	

Table 67. Test Case Id: TC_C_04_CS

Test case name	**Local Stop Transaction - Different idToken**	
Test case Id	TC_C_04_CS	
Use case Id(s)	C01, C04, E07	
Requirement(s)	C01.FR.02, C01.FR.03	
System under test	Charging Station	
Description	The EV Driver tries to stop an ongoing transaction, by locally presenting a different IdToken.	
Purpose	To verify whether the Charging Station does not stop the charging session when a different idToken is presented, than the one used to start the transaction.	
Prerequisite(s)	- The Charging Station supports at least one authorization method described at the following Use cases; C01, C04. - The Charging Station does NOT use one idToken reader for multiple EVSE. - The Charging Station supports authorization methods other than NoAuthorization	
Before (Preparations)	**Configuration State:** **AuthCtrlr.AuthEnabled** is *true* (If implemented AND ReadWrite) **AuthCtrlr.DisableRemoteAuthorization** is *false* (If implemented)	
	Memory State: - The "different idToken" does not exist in Authorization Cache or Local Authorization List. - The "different idToken" does not have an associated GroupId that matches with the GroupId of the "starting idToken".	
	Reusable State(s): **State is** *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Present a different idToken than used to start the transaction.*	
	1. The Charging Station sends an **AuthorizeRequest**	**2.** The OCTT responds with an **AuthorizeResponse** with **idTokenInfo.status** *Accepted*
	Note(s): *- The Charging Station SHALL NOT send a TransactionEventRequest message with an **idToken** field after receiving an idToken that is different, than the one used to start the transaction.* *- The OCTT waits <Configured message timeout> seconds, before ending the testcase.*	
Tool validations	N/a	
	Post scenario validations: - Charging Station has not sent a TransactionEventRequest(*Ended*).	

Table 68. Test Case Id: TC_C_06_CS

Test case name	**Local start transaction - Authorization Blocked**	
Test case Id	TC_C_06_CS	
Use case Id(s)	C01	
Requirement(s)	C01.FR.02	
System under test	Charging Station	
Description	When a Charging Station needs to charge an EV, it needs to authorize the EV Driver first at the CSMS before the charging can be started or stopped.	
Purpose	To verify whether the Charging Station is able to handle receiving an Blocked idToken.	
Prerequisite(s)	The Charging Station does NOT have a cable lock, which prevents the EV driver to connect the EV and EVSE before authorization. The Charging Station supports authorization methods other than NoAuthorization	
Before (Preparations)	**Configuration State:** **AuthCtrlr.AuthEnabled** is *true* (If implemented AND ReadWrite) **AuthCtrlr.DisableRemoteAuthorization** is *false* (If implemented)	
	Memory State: N/a	
	Reusable State(s): **State is** *EVConnectedPreSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Present idToken.*	
	1. The Charging Station sends an **AuthorizeRequest**	**2.** The OCTT responds with an **AuthorizeResponse** with **idTokenInfo.status** *Blocked*
	Note(s): *- The Charging Station SHALL NOT send a TransactionEventRequest message after the AuthorizeRequest from step 7.* *- The OCTT waits <Configured message timeout> seconds, before ending the testcase.*	
Tool validations	* Step 1: Message: **AuthorizeRequest** - **idToken.idToken** *<Configured blocked_idtoken_idtoken>* - **idToken.type** *<Configured blocked_idtoken_type>*	
	Post scenario validations: N/a	

Table 69. Test Case Id: TC_C_07_CS

Test case name	**Local start transaction - Authorization Expired**	
Test case Id	TC_C_07_CS	
Use case Id(s)	C01	
Requirement(s)	C01.FR.02	
System under test	Charging Station	
Description	When a Charging Station needs to charge an EV, it needs to authorize the EV Driver first at the CSMS before the charging can be started or stopped.	
Purpose	To verify whether the Charging Station is able to handle receiving an Expired idToken.	
Prerequisite(s)	The Charging Station does NOT have a cable lock, which prevents the EV driver to connect the EV and EVSE before authorization. The Charging Station supports authorization methods other than NoAuthorization	
Before (Preparations)	**Configuration State:** **AuthCtrlr.AuthEnabled** is *true* (If implemented AND ReadWrite) **AuthCtrlr.DisableRemoteAuthorization** is *false* (If implemented)	
	Memory State: N/a	
	Reusable State(s): **State is** *EVConnectedPreSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Present idToken.*	
	1. The Charging Station sends an **AuthorizeRequest**	**2.** The OCTT responds with an **AuthorizeResponse** with **idTokenInfo.status** *Expired*
	Note(s): *- The Charging Station SHALL NOT send a TransactionEventRequest message after the AuthorizeRequest from step 7.* *- The OCTT waits <Configured message timeout> seconds, before ending the testcase.*	
Tool validations	* Step 1: Message: **AuthorizeRequest** - **idToken.idToken** *<Configured expired_idtoken_idtoken>* - **idToken.type** *<Configured expired_idtoken_type>*	
	Post scenario validations: N/a	

Table 70. Test Case Id: TC_C_08_CS

Test case name	**Authorization through authorization cache - Accepted**	
Test case Id	TC_C_08_CS	
Use case Id(s)	C12	
Requirement(s)	C12_FR_02, C12_FR_04	
System under test	Charging Station	
Description	This test case describes how the EV Driver is authorized to start a transaction while the Charging Station uses Cached IdToken. This enables the EV Driver to Online start a transaction by using the Authorization Cache in which the Charging Station can respond faster, as no AuthorizeRequest is being sent.	
Purpose	To verify if the Charging Station is able to Authorize an idToken which has status "Accepted" in its cache according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	- AuthCacheCtrlr.AuthCacheAvailable is implemented with value true - The Charging Station supports authorization methods other than NoAuthorization	
Before (Preparations)	**Configuration State:** **AuthCacheCtrlr.AuthCacheEnabled** is *true* (If implemented) **AuthCtrlr.LocalPreAuthorize** is *true* (If implemented)	
	Memory State: *IdTokenCached* for <Configured valid IdToken fields>	
	Reusable State(s): **State is** *EVConnectedPreSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *Authorized*	
	Note(s): *Present valid idToken which is already configured in the Authorization Cache*	
	2. Execute **Reusable State** *EnergyTransferStarted*	
Tool validations	N/a	
	Post scenario validations: - Energy transfer is started	

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

Table 71. Test Case Id: TC_C_09_CS

Test case name	**Authorization through authorization cache - Invalid & Not Accepted**	
Test case Id	TC_C_09_CS	
Use case Id(s)	C12	
Requirement(s)	C12_FR_05, C10_FR_03	
System under test	Charging Station	
Description	This test case describes how the EV Driver is authorized to start a transaction while the Charging Station uses Cached IdToken. This enables the EV Driver to Online start a transaction by using the Authorization Cache in which the Charging Station can respond faster, as no AuthorizeRequest is being sent.	
Purpose	To verify if the Charging Station is able to Authorize an idToken which has status "Invalid" in its cache according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	AuthCacheCtrlr.AuthCacheAvailable is implemented with value true The Charging Station supports authorization methods other than NoAuthorization	
Before (Preparations)	**Configuration State:** **AuthCacheCtrlr.AuthCacheEnabled** is *true* (If implemented) **AuthCtrlr.LocalPreAuthorize** is *true* (If implemented) **AuthCacheCtrlrDisablePostAuthorize** is *false* (If implemented)	
	Memory State: *IdTokenCached* for <Configured invalid IdToken fields>	
	Reusable State(s): **State is** *EVConnectedPreSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Present Invalid idToken which is already configured in the Authorization Cache*	
	1. The Charging Station sends a **AuthorizeRequest**	**2.** The OCTT responds with a **AuthorizeResponse** with **idTokenInfo.status** *Invalid*
Tool validations	* Step 1: Message **AuthorizeRequest** - **idToken.idToken** *<Configured invalid_idtoken_idtoken>* - **idToken.type** *<Configured invalid_idtoken_type>*	
	Post scenario validations: - N/a	

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

Table 72. Test Case Id: TC_C_10_CS

Test case name	**Authorization through authorization cache - Blocked**	
Test case Id	TC_C_10_CS	
Use case Id(s)	C12	
Requirement(s)	C12_FR_05, C10_FR_03	
System under test	Charging Station	
Description	This test case describes how the EV Driver is authorized to start a transaction while the Charging Station uses Cached IdToken. This enables the EV Driver to Online start a transaction by using the Authorization Cache in which the Charging Station can respond faster, as no AuthorizeRequest is being sent.	
Purpose	To verify if the Charging Station is able to Authorize an idToken which has status "Blocked" in its cache according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	AuthCacheCtrlr.AuthCacheAvailable is implemented with value true The Charging Station supports authorization methods other than NoAuthorization	
Before (Preparations)	**Configuration State:** **AuthCacheCtrlr.AuthCacheEnabled** is *true* (If implemented) **AuthCtrlr.LocalPreAuthorize** is *true* (If implemented) **AuthCacheCtrlrDisablePostAuthorize** is *false* (If implemented)	
	Memory State: *IdTokenCached* for <Configured blocked IdToken fields>	
	Reusable State(s): **State is** *EVConnectedPreSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Present Blocked idToken which is already configured in the Authorization Cache*	
	1. The Charging Station sends a **AuthorizeRequest**	**2.** The OCTT responds with a **AuthorizedResponse** with **idTokenInfo.status** *Blocked*
Tool validations	* Step 1: Message **AuthorizeRequest** - **idToken.idToken** *<Configured blocked_idtoken_idtoken>* - **idToken.type** *<Configured blocked_idtoken_type>*	
	Post scenario validations: - N/a	

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

Table 73. Test Case Id: TC_C_11_CS

Test case name	**Authorization through authorization cache - Expired**	
Test case Id	TC_C_11_CS	
Use case Id(s)	C12	
Requirement(s)	C12_FR_05, C10_FR_03	
System under test	Charging Station	
Description	This test case describes how the EV Driver is authorized to start a transaction while the Charging Station uses Cached IdToken. This enables the EV Driver to Online start a transaction by using the Authorization Cache in which the Charging Station can respond faster, as no AuthorizeRequest is being sent.	
Purpose	To verify if the Charging Station is able to Authorize an idToken which has status "Expired" in its cache according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	AuthCacheCtrlr.AuthCacheAvailable is implemented with value true The Charging Station supports authorization methods other than NoAuthorization	
Before (Preparations)	**Configuration State:** **AuthCacheCtrlr.AuthCacheEnabled** is *true* (If implemented) **AuthCtrlr.LocalPreAuthorize** is *true* (If implemented) **AuthCacheCtrlrDisablePostAuthorize** is *false* (If implemented)	
	Memory State: *IdTokenCached* for <Configured expired IdToken fields>	
	Reusable State(s): **State is** *EVConnectedPreSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Present Expired idToken which is already configured in the Authorization Cache*	
	1. The Charging Station sends a **AuthorizeRequest**	**2.** The OCTT responds with a **TransactionEventResponse** with **idTokenInfo.status** *Expired*
Tool validations	* Step 1: Message **AuthorizeRequest** - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>*	
	Post scenario validations: - N/a	

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

Table 74. Test Case Id: TC_C_12_CS

Test case name	**Authorization through authorization cache - Invalid & Accepted**	
Test case Id	TC_C_12_CS	
Use case Id(s)	C12	
Requirement(s)	C12_FR_05, C10_FR_03	
System under test	Charging Station	
Description	This test case describes how the EV Driver is authorized to start a transaction while the Charging Station uses Cached IdToken. This enables the EV Driver to Online start a transaction by using the Authorization Cache in which the Charging Station can respond faster, as no AuthorizeRequest is being sent.	
Purpose	To verify if the Charging Station is able to Authorize an idToken which has status "Invalid" in its cache, but the CSMS has status "Valid", according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	- AuthCacheCtrlr.AuthCacheAvailable is implemented with value true - The Charging Station supports authorization methods other than NoAuthorization	
Before (Preparations)	**Configuration State:** **AuthCacheCtrlr.AuthCacheEnabled** is *true* (If implemented) **AuthCtrlr.LocalPreAuthorize** is *true* (If implemented) **AuthCacheCtrlrDisablePostAuthorize** is *false* (If implemented)	
	Memory State: *IdTokenCached* for <Configured invalid IdToken fields>	
	Reusable State(s): **State is** *EVConnectedPreSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Present Invalid idToken which is already configured in the Authorization Cache*	
	1. The Charging Station sends an **AuthorizeRequest**	**2.** The OCTT responds with an **AuthorizeResponse** with **idTokenInfo.status** *Accepted*
	3. The Charging Station sends a **TransactionEventRequest** Note(s): *-This step is optional.*	**4.** The OCTT responds with a **TransactionEventResponse**
	5. Execute **Reusable State** *EnergyTransferStarted*	
Tool validations	* Step 1: Message **AuthorizeRequest** - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>*	
	Post scenario validations: - Energy transfer is started	

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

Table 75. Test Case Id: TC_C_13_CS

Test case name	**Authorization through authorization cache - Accepted but cable not connected yet.**	
Test case Id	TC_C_13_CS	
Use case Id(s)	C12	
Requirement(s)	C12_FR_02, C12_FR_04	
System under test	Charging Station	
Description	This test case describes how the EV Driver is authorized to start a transaction while the Charging Station uses Cached IdToken. This enables the EV Driver to Online start a transaction by using the Authorization Cache in which the Charging Station can respond faster, as no AuthorizeRequest is being sent.	
Purpose	To verify if the Charging Station is able to Authorize an idToken which has status "Accepted" in its cache but the cable is not connected yet according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	- AuthCacheCtrlr.AuthCacheAvailable is implemented with value true - The Charging Station supports authorization methods other than NoAuthorization	
Before (Preparations)	**Configuration State:** **AuthCacheCtrlr.AuthCacheEnabled** is *true* (If implemented) **AuthCtrlr.LocalPreAuthorize** is *true* (If implemented)	
	Memory State: *IdTokenCached* for <Configured valid IdToken fields> (If implemented)	
	Reusable State(s): If applicable, **State is** *ParkingBayOccupied*	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Present valid idToken which is already configured in the Authorization Cache*	
	1. The Charging Station sends a **TransactionEventRequest** Note(s): *- This step needs to be executed when **TxStartPoint** contains Authorized OR the transaction already started. So in the case **TxStartPoint** contains ParkingBayOccupancy. - As long as the cable is not plugged in the energy transfer will not start.*	**2.** The OCTT responds with a **TransactionEventResponse** with **idTokenInfo.status** *Accepted*
	3. Execute **Reusable State** *EVConnectedPreSession*	
	4. Execute **Reusable State** *EnergyTransferStarted*	
Tool validations	* Step 1: Message **TransactionEventRequest** - **triggerReason** *Authorized* - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>* if transaction was already started - **eventType** *Updated* else - **eventType** *Started*	
	Post scenario validations: - Energy transfer is started	

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

Table 76. Test Case Id: TC_C_14_CS

Test case name	**Authorization through authorization cache - GroupID equal to MasterPassGroupId.**	
Test case Id	TC_C_14_CS	
Use case Id(s)	C12	
Requirement(s)	C12_FR_09, C16.FR.03	
System under test	Charging Station	
Description	This test case describes how the EV Driver is authorized to start a transaction while the Charging Station uses Cached IdToken. This enables the EV Driver to Online start a transaction by using the Authorization Cache in which the Charging Station can respond faster, as no AuthorizeRequest is being sent.	
Purpose	To verify if the Charging Station is able to respond correctly to an idToken which has the "MasterPassGroupId" as group id according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	AuthCacheCtrlr.AuthCacheAvailable is implemented with value true The Charging station supports MasterPass feature. The Charging Station supports authorization methods other than NoAuthorization	
Before (Preparations)	**Configuration State:** **AuthCacheCtrlr.AuthCacheEnabled** is *true* (If implemented) **AuthCtrlr.LocalPreAuthorize** is *true* (If implemented) **AuthCtrlr.MasterPassGroupId** is *<Configured MasterPassGroupId>*	
	Memory State: Store the idToken that is part of the MasterPass group at the cache.	
	Reusable State(s): **State is** *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Present the idToken with group id "MasterPassGroupId" which is already configured in the Authorization Cache*	
	1. The Charging Station sends a **TransactionEventRequest**	**2.** The OCTT responds with a **TransactionEventResponse**
	3. Execute **Reusable State** *EVConnectedPostSession*	
	4. Execute **Reusable State** *EVDisconnected*	
	5. Execute **Reusable State** *ParkingBayUnoccupied*	
	Note(s): *Step 3, 4, and 5 are only executed if the transaction is still running.*	
Tool validations	* Step 2: Message **TransactionEventRequest** - **triggerReason** *StopAuthorized* - **idToken.idToken** *<Configured masterPass idToken>* - **idToken.type** *<Configured masterPass idTokenType>* If **eventType** *Ended*, then: - **transactionInfo.stoppedReason** *MasterPass*	
	Post scenario validations: - N/a	

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

Table 77. Test Case Id: TC_C_15_CS

Test case name	**Authorization through authorization cache - StopTxOnInvalidId = false, MaxEnergyOnInvalidId > 0**	
Test case Id	TC_C_15_CS	
Use case Id(s)	C12	
Requirement(s)	C12_FR_02, C12_FR_04	
System under test	Charging Station	
Description	This test case describes how the EV Driver is authorized to start a transaction while the Charging Station uses Cached IdToken. This enables the EV Driver to Online start a transaction by using the Authorization Cache in which the Charging Station can respond faster, as no AuthorizeRequest is being sent.	
Purpose	To verify if the Charging Station is able to respond correctly to an idToken which has status "Accepted" in its cache but is "Invalid" in the CSMS with certain values of StopTxOnInvalidId and MaxEnergyOnInvalidId according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	- AuthCacheCtrlr.AuthCacheAvailable is implemented with value true - The Charging Station has MaxEnergyOnInvalidId implemented - At least one of the following must be supported; Local auth list, auth cache, StartTxUnknownIds. - The Charging Station supports authorization methods other than NoAuthorization	
Before (Preparations)	**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>*	
	Memory State: *IdTokenCached* for <Configured valid IdToken fields> (If implemented) A known valid idToken is configured in the Local auth list (if implemented) and auth cache (if implemented)	
	Reusable State(s): **State is** *EVConnectedPreSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. *The OCTT closes the WebSocket connection AND does not accept a reconnect.*	
	Manual Action: *Present valid idToken which is already configured in the Authorization Cache*	
	Note(s): *The OCTT will wait for _<Configured Transaction Duration>* seconds_	
	2. *The OCTT accepts reconnection attempt from the Charging Station.*	
	Note(s): *- The Charging Station will empty its Transaction message queue. This will contain one or more TransactionEventRequest messages*	
	3. The Charging Station sends a **TransactionEventRequest**	**4.** The OCTT responds with a **TransactionEventResponse** with **idTokenInfo.status** *Invalid* (if idToken is not omitted)

Test case name	**Authorization through authorization cache - StopTxOnInvalidId = false, MaxEnergyOnInvalidId > 0**
Tool validations	* 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*
	Post scenario validations: - Energy transfer is started but only MaxEnergyOnInvalidId amount of energy is delivered

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

Table 78. Test Case Id: TC_C_16_CS

Test case name	**Authorization through authorization cache - StopTxOnInvalidId = true**	
Test case Id	TC_C_16_CS	
Use case Id(s)	C12	
Requirement(s)	C12_FR_02, C12_FR_04	
System under test	Charging Station	
Description	This test case describes how the EV Driver is authorized to start a transaction while the Charging Station uses Cached IdToken. This enables the EV Driver to Online start a transaction by using the Authorization Cache in which the Charging Station can respond faster, as no AuthorizeRequest is being sent.	
Purpose	To verify if the Charging Station is able to respond correctly to an idToken which has status "Accepted" in its cache but is "Invalid" in the CSMS when StopTxOnInvalidId is true according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	- AuthCacheCtrlr.AuthCacheAvailable is implemented with value true - At least one of the following must be supported; Local auth list, auth cache, StartTxUnknownIds. - The Charging Station supports authorization methods other than NoAuthorization	
Before (Preparations)	**Configuration State:** **AuthCacheEnabled** is *true* (If implemented) **LocalPreAuthorize** is *true* (If implemented) **LocalAuthorizeOffline** is *true* **StopTxOnInvalidId** is *true* **MaxEnergyOnInvalidId** is *0*	
	Memory State: *IdTokenCached* for <Configured valid IdToken fields> (If implemented) A known valid idToken is configured in the Local auth list (if implemented) and auth cache (if implemented)	
	Reusable State(s): **State is** *EVConnectedPreSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. *The OCTT closes the WebSocket connection AND does not accept a reconnect.*	
	Manual Action: *Present valid idToken which is already configured in the Authorization Cache*	
	Note(s): *The OCTT will wait for 5 seconds*	
	2. *The OCTT accepts reconnection attempt from the Charging Station.*	
	Note(s): *- The Charging Station will empty its Transaction message queue. This will contain one or more TransactionEventRequest messages*	
	3. The Charging Station sends a **TransactionEventRequest**	**4.** The OCTT responds with a **TransactionEventResponse** with **idTokenInfo.status** *Invalid* (if idToken is not omitted)
	5. The Charging Station sends a **TransactionEventRequest** with **triggerReason** *Deauthorized*	**6.** The OCTT responds with a **TransactionEventResponse**

Test case name	**Authorization through authorization cache - StopTxOnInvalidId = true**
Tool validations	* 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* A message with: - **triggerReason** *Deauthorized*
	Post scenario validations: - Energyflow stops on receiving status invalid

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

Table 79. Test Case Id: TC_C_17_CS

Test case name	**Authorization through authorization cache - StopTxOnInvalidId = false**	
Test case Id	TC_C_17_CS	
Use case Id(s)	C12	
Requirement(s)	C12_FR_02, C12_FR_04	
System under test	Charging Station	
Description	This test case describes how the EV Driver is authorized to start a transaction while the Charging Station uses Cached IdToken. This enables the EV Driver to Online start a transaction by using the Authorization Cache in which the Charging Station can respond faster, as no AuthorizeRequest is being sent.	
Purpose	To verify if the Charging Station is able to respond correctly to an idToken which has status "Accepted" in its cache but is "Invalid" in the CSMS when StopTxOnInvalidId is false according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	- AuthCacheCtrlr.AuthCacheAvailable is implemented with value true - At least one of the following must be supported; Local auth list, auth cache, StartTxUnknownIds. - The Charging Station supports authorization methods other than NoAuthorization	
Before (Preparations)	**Configuration State:** **AuthCacheEnabled** is *true* (If implemented) **LocalPreAuthorize** is *true* (If implemented) **OfflineTxForUnknownIdEnabled** is *true* (If implemented) **StopTxOnInvalidId** is *false* **MaxEnergyOnInvalidId** is *0*	
	Memory State: *IdTokenCached* for <Configured valid IdToken fields> (If implemented) A known valid idToken is configured in the Local auth list (if implemented) and auth cache (if implemented)	
	Reusable State(s): **State is** *EVConnectedPreSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. *The OCTT closes the WebSocket connection AND does not accept a reconnect.*	
	Manual Action: *Present valid idToken which is already configured in the Authorization Cache*	
	Note(s): *The OCTT will wait for 5 seconds*	
	2. *The OCTT accepts reconnection attempt from the Charging Station.*	
	Note(s): - *The Charging Station will empty its Transaction message queue. This will contain one or more TransactionEventRequest messages*	
	3. The Charging Station sends a **TransactionEventRequest**	**4.** The OCTT responds with a **TransactionEventResponse** with **idTokenInfo.status** *Invalid* (if idToken is not omitted)
	5. The Charging Station sends a **TransactionEventRequest** with **triggerReason** *SuspendedEVSE*	**6.** The OCTT responds with a **TransactionEventResponse**
Tool validations	* Step 2: Message **TransactionEventRequest** A message with: - **triggerReason** *Authorized* - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>* - **offline** *True* A message with: - **transactionInfo.chargingState** *SuspendedEVSE* No message with: - **triggerReason** *SuspendedEVSE*	
	Post scenario validations: - Energyflow stops on receiving status invalid	

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

Table 80. Test Case Id: TC_C_18_CS

Test case name	**Authorization through authorization cache - StopTxOnInvalidId = true, MaxEnergyOnInvalidId > 0**	
Test case Id	TC_C_18_CS	
Use case Id(s)	C12	
Requirement(s)	C12_FR_02, C12_FR_04	
System under test	Charging Station	
Description	This test case describes how the EV Driver is authorized to start a transaction while the Charging Station uses Cached IdToken. This enables the EV Driver to Online start a transaction by using the Authorization Cache in which the Charging Station can respond faster, as no AuthorizeRequest is being sent.	
Purpose	To verify if the Charging Station is able to respond correctly to an idToken which has status "Accepted" in its cache but is "Invalid" in the CSMS when StopTxOnInvalidId is true and MaxEnergyOnInvalidId > 0 according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	- AuthCacheCtrlr.AuthCacheAvailable is implemented with value true - The Charging Station has MaxEnergyOnInvalidId implemented. - At least one of the following must be supported; Local auth list, auth cache, StartTxUnknownIds. - The Charging Station supports authorization methods other than NoAuthorization	
Before (Preparations)	**Configuration State:** **AuthCacheEnabled** is *true* (If implemented) **LocalPreAuthorize** is *true* (If implemented) **LocalAuthorizeOffline** is *true* **OfflineTxForUnknownIdEnabled** is *true* (If implemented) **StopTxOnInvalidId** is *true* **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>*	
	Memory State: *IdTokenCached* for <Configured valid IdToken fields> (If implemented) A known valid idToken is configured in the Local auth list (if implemented) and auth cache (if implemented)	
	Reusable State(s): **State is** *EVConnectedPreSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. *The OCTT closes the WebSocket connection AND does not accept a reconnect.*	
	Manual Action: *Present valid idToken which is already configured in the Authorization Cache*	
	Note(s): *The OCTT will wait for _<Configured Transaction Duration>* seconds_	
	2. *The OCTT accepts reconnection attempt from the Charging Station.*	
	Note(s): *- The Charging Station will empty its Transaction message queue. This will contain one or more TransactionEventRequest messages*	
	3. The Charging Station sends a **TransactionEventRequest**	**4.** The OCTT responds with a **TransactionEventResponse** with **idTokenInfo.status** *Invalid* (if idToken is not omitted)
	5. The Charging Station sends a **TransactionEventRequest** with **triggerReason** *Deauthorized*	**6.** The OCTT responds with a **TransactionEventResponse**

Test case name	**Authorization through authorization cache - StopTxOnInvalidId = true, MaxEnergyOnInvalidId > 0**
Tool validations	* 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* * Step 5: A message with: - **triggerReason** *Deauthorized* - **offline** *False*
	Post scenario validations: - Energyflow stops on receiving status invalid

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

Table 81. Test Case Id: TC_C_57_CS

Test case name	**Authorization through authorization cache - AuthCacheDisablePostAuthorize**	
Test case Id	TC_C_57_CS	
Use case Id(s)	C12	
Requirement(s)	C12_FR_05, C10_FR_03	
System under test	Charging Station	
Description	This test case describes how the EV Driver can be authorized to start a transaction by using Cached IdTokens. This enables the EV Driver to start a transaction while the Charging Station is online by using the Authorization Cache in which case the Charging Station can respond faster, since no AuthorizeRequest is being sent. For an IdToken that does not have status "Accepted" the Charging Station will still send an AuthorizeRequest to check against the most recent status from CSMS. However, when the setting AuthCacheDisablePostAuthorize is set to true, then the Charging Station will not do this.	
Purpose	To verify that the Charging Station will not send an AutorizeRequest for an IdToken in the Authorization Cache that is not "Accepted", when AuthCacheDisablePostAuthorize is set to true.	
Prerequisite(s)	- AuthCacheCtrlr.Available is implemented with value *true* - AuthCacheCtrlr.DisablePostAuthorize is implemented - The Charging Station supports authorization methods other than NoAuthorization	
Before (Preparations)	**Configuration State:** **AuthCacheCtrlr.Enabled** is *true* (If implemented) **AuthCtrlr.LocalPreAuthorize** is *true* (If implemented) **AuthCacheCtrlr.DisablePostAuthorize** is *true*	
	Memory State: *IdTokenCached* for <Configured invalid IdToken fields>	
	Reusable State(s): **State is** *EVConnectedPreSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Present Invalid idToken which is already configured in the Authorization Cache*	
	1. The Charging Station does NOT send a **AuthorizeRequest**	
Tool validations	* Step 1: Check that Charging Station does NOT send an AuthorizeRequest and authorization is refused.	
	Post scenario validations: - N/a	

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

Table 82. Test Case Id: TC_C_27_CS

Test case name	**Online authorization through local authorization list - Accepted**	
Test case Id	TC_C_27_CS	
Use case Id(s)	C14	
Requirement(s)	C14_FR_02	
System under test	Charging Station	
Description	This test case describes how to authorize an IdToken via the Local Authorization List while the Charging Station is online. When online the Charging Station can then locally authorize the IdToken, and is not required to send an AuthorizeRequest for a known IdToken.	
Purpose	To verify if the Charging Station is able to authorize an idToken which has status "Accepted" in its local authorization list according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	- LocalAuthListCtrlr.LocalAuthListAvailable is implemented with value true - The Charging Station supports authorization methods other than NoAuthorization	
Before (Preparations)	**Configuration State:** **LocalAuthListEnabled** is *true* (If implemented) **LocalPreAuthorize** is *true* (If implemented)	
	Memory State: A known valid idToken is configured in the Local Authorization List	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Present valid idToken which is already configured in the Local Authorization List*	
	1. The Charging Station sends a **TransactionEventRequest** Note(s): *- This step needs to be executed when **TxStartPoint** contains Authorized OR the transaction already started. So in the case **TxStartPoint** contains ParkingBayOccupancy*	**2.** The OCTT responds with a **TransactionEventResponse** with - **idTokenInfo.status** *Accepted*
	3. Execute **Reusable State** *EVConnectedPreSession*	
	4. Execute **Reusable State** *EnergyTransferStarted*	
Tool validations	* Step 1: Message **TransactionEventRequest** - **triggerReason** *Authorized* - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>*	
	Post scenario validations: - Energy is transferred	

Table 83. Test Case Id: TC_C_28_CS

Test case name	**Online authorization through local authorization list - Invalid & Not Accepted**	
Test case Id	TC_C_28_CS	
Use case Id(s)	C14	
Requirement(s)	C14_FR_03	
System under test	Charging Station	
Description	This test case describes how to authorize an IdToken via the Local Authorization List while the Charging Station is online. When online the Charging Station can then locally authorize the IdToken, and is not required to send an AuthorizeRequest for a known IdToken.	
Purpose	To verify if the Charging Station is able to respond correctly to an idToken which has status "Invalid" in its local authorization list according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	- LocalAuthListCtrlr.LocalAuthListAvailable is implemented with value true - The Charging Station supports authorization methods other than NoAuthorization	
Before (Preparations)	**Configuration State:** **LocalAuthListEnabled** is *true* (If implemented) **LocalPreAuthorize** is *true* (If implemented) *LocalAuthListDisablePostAuthorize * *false* (If implemented)	
	Memory State: A known invalid idToken is configured in the Local Authorization List	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Present invalid idToken which is already configured in the Local Authorization List*	
	1. The Charging Station sends a **AuthorizeRequest**	**2.** The OCTT responds with a **AuthorizeResponse** with **idTokenInfo.status** *Invalid*
Tool validations	* Step 1: Message **AuthorizeRequest** - **idToken.idToken** *<Configured invalid_idtoken_idtoken>* - **idToken.type** *<Configured invalid_idtoken_type>*	
	Post scenario validations: - N/a	

Table 84. Test Case Id: TC_C_29_CS

Test case name	**Online authorization through local authorization list - Blocked**	
Test case Id	TC_C_29_CS	
Use case Id(s)	C14	
Requirement(s)	C14_FR_03	
System under test	Charging Station	
Description	This test case describes how to authorize an IdToken via the Local Authorization List while the Charging Station is online. When online the Charging Station can then locally authorize the IdToken, and is not required to send an AuthorizeRequest for a known IdToken.	
Purpose	To verify if the Charging Station is able to respond correctly to an idToken which has status "Blocked" in its local authorization list according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	- LocalAuthListCtrlr.LocalAuthListAvailable is implemented with value true - The Charging Station supports authorization methods other than NoAuthorization	
Before (Preparations)	**Configuration State:** **LocalAuthListEnabled** is *true* (If implemented) **LocalPreAuthorize** is *true* (If implemented) *LocalAuthListDisablePostAuthorize * *false* (If implemented)	
	Memory State: A known blocked idToken is configured in the Local Authorization List	
	Reusable State(s): **State is** *EVConnectedPreSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Present blocked idToken which is already configured in the Local Authorization List*	
	1. The Charging Station sends a **AuthorizeRequest**	**2.** The OCTT responds with a **AuthorizeResponse** with **idTokenInfo.status** *Blocked*
Tool validations	* Step 1: Message **AuthorizeRequest** - **idToken.idToken** *<Configured blocked_idtoken_idtoken>* - **idToken.type** *<Configured blocked_idtoken_type>*	
	Post scenario validations: - N/a	

Table 85. Test Case Id: TC_C_30_CS

Test case name	**Online authorization through local authorization list - Expired**	
Test case Id	TC_C_30_CS	
Use case Id(s)	C14	
Requirement(s)	C14_FR_03	
System under test	Charging Station	
Description	This test case describes how to authorize an IdToken via the Local Authorization List while the Charging Station is online. When online the Charging Station can then locally authorize the IdToken, and is not required to send an AuthorizeRequest for a known IdToken.	
Purpose	To verify if the Charging Station is able to respond correctly to an idToken which has status "Expired" in its local authorization list according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	- LocalAuthListCtrlr.LocalAuthListAvailable is implemented with value true - The Charging Station supports authorization methods other than NoAuthorization	
Before (Preparations)	**Configuration State:** **LocalAuthListEnabled** is *true* (If implemented) **LocalPreAuthorize** is *true* (If implemented) *LocalAuthListDisablePostAuthorize * *false* (If implemented)	
	Memory State: A known expired idToken is configured in the Local Authorization List	
	Reusable State(s): **State is** *EVConnectedPreSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Present expired idToken which is already configured in the Local Authorization List*	
	1. The Charging Station sends a **AuthorizeRequest**	**2.** The OCTT responds with a **AuthorizeResponse** with **idTokenInfo.status** *Expired*
Tool validations	* Step 1: Message **AuthorizeRequest** - **idToken.idToken** *<Configured expired_idtoken_idtoken>* - **idToken.type** *<Configured expired_idtoken_type>*	
	Post scenario validations: - N/a	

Table 86. Test Case Id: TC_C_31_CS

Test case name	**Online authorization through local authorization list - Invalid & Accepted**	
Test case Id	TC_C_31_CS	
Use case Id(s)	C14	
Requirement(s)	C14_FR_03	
System under test	Charging Station	
Description	This test case describes how to authorize an IdToken via the Local Authorization List while the Charging Station is online. When online the Charging Station can then locally authorize the IdToken, and is not required to send an AuthorizeRequest for a known IdToken.	
Purpose	To verify if the Charging Station is able to respond correctly to an idToken which has status "Invalid" in its local authorization list, but is actually valid for the CSMS, according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	- LocalAuthListCtrlr.LocalAuthListAvailable is implemented with value true - The Charging Station supports authorization methods other than NoAuthorization	
Before (Preparations)	**Configuration State:** **LocalAuthListEnabled** is *true* (If implemented) **LocalPreAuthorize** is *true* (If implemented) *LocalAuthListDisablePostAuthorize * *false* (If implemented)	
	Memory State: A known invalid idToken is configured in the Local Authorization List	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *Authorized* Note(s): - *Present invalid idToken which is already configured in the Local Authorization List*	
	2. Execute **Reusable State** *EVConnectedPreSession*	
	3. Execute **Reusable State** *EnergyTransferStarted*	
Tool validations	- N/a	
	Post scenario validations: - Energy is transferred	

Table 87. Test Case Id: TC_C_58_CS

Test case name	**Online authorization through local authorization list - LocalAuthListDisablePostAuthorize**	
Test case Id	TC_C_28_CS	
Use case Id(s)	C14	
Requirement(s)	C14_FR_03	
System under test	Charging Station	
Description	This test case describes how to authorize an IdToken via the Local Authorization List while the Charging Station is online. While online the Charging Station can then locally authorize the IdToken, and is not required to send an AuthorizeRequest for a known IdToken. For an IdToken that does not have status "Accepted" the Charging Station will still send an AuthorizeRequest to check against the most recent status from CSMS. However, when the setting LocalAuthListDisablePostAuthorize is set to true, then the Charging Station will not do this.	
Purpose	To verify that the Charging Station will not send an AuthorizeRequest for an idToken which has status "Invalid" in its local authorization list.	
Prerequisite(s)	- LocalAuthListCtrlr.Available is implemented with value *true* - LocalAuthListCtrlr.DisablePostAuthorize is implemented. - The Charging Station supports authorization methods other than NoAuthorization	
Before (Preparations)	**Configuration State:** **LocalAuthListCtrlr.Enabled** is *true* (If implemented) **AuthCtrlr.LocalPreAuthorize** is *true* (If implemented) **LocalAuthListCtrlr.DisablePostAuthorize** *true*	
	Memory State: *IdTokenLocalAuthList* for *<Configured invalid idtoken fields>*	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Present invalid idToken which is already configured in the Local Authorization List*	
	1. The Charging Station does NOT send a **AuthorizeRequest**	
Tool validations	* Step 1: Check that Charging Station does NOT send an AuthorizeRequest and authorization is refused.	
	Post scenario validations: - N/a	

Table 88. Test Case Id: TC_C_32_CS

Test case name	**Store Authorization Data in the Authorization Cache - Persistent over reboot**	
Test case Id	TC_C_32_CS	
Use case Id(s)	C10	
Requirement(s)	C10_FR_02	
System under test	Charging Station	
Description	This test case covers how the Charging Station autonomously stores a record of previously presented identifiers that have been successfully authorized by the CSMS in the Authorization Cache. (Successfully meaning: a response received on a message containing an IdToken)	
Purpose	To verify if the Charging Station is able to store the identifiers persistent over reboot according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	- The Charging Station supports the Authorization Cache feature - Authorization cache is stored in the non-volatile memory. - The Charging Station supports authorization methods other than NoAuthorization	
Before (Preparations)	**Configuration State:** **AuthCacheEnabled** is *true* (If implemented) **LocalPreAuthorize** is *true*	
	Memory State: *IdTokenCached* for <Configured valid IdToken fields>	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *Booted*	
	Manual Action: *Present valid idToken which is already configured in the Authorization Cache*	
	2. The Charging Station sends a **TransactionEventRequest** Note(s): *- This step needs to be executed when **TxStartPoint** contains Authorized OR the transaction already started. So in the case **TxStartPoint** contains ParkingBayOccupancy. - As long as the cable is not plugged in the energy transfer will not start.*	**3.** The OCTT responds with a **TransactionEventResponse** with **idTokenInfo.status** *Accepted*
	4. Execute **Reusable State** *EVConnectedPreSession*	
	5. Execute **Reusable State** *EnergyTransferStarted*	
Tool validations	* Step 2: Message **TransactionEventRequest** - **triggerReason** *Authorized* - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>*	
	Post scenario validations: - N/a	

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

Table 89. Test Case Id: TC_C_33_CS

Test case name	**Store Authorization Data in the Authorization Cache - Update on AuthorizeResponse**	
Test case Id	TC_C_33_CS	
Use case Id(s)	C10	
Requirement(s)	C10_FR_04, C12.FR.06	
System under test	Charging Station	
Description	This test case covers how the Charging Station autonomously stores a record of previously presented identifiers that have been successfully authorized by the CSMS in the Authorization Cache. (Successfully meaning: a response received on a message containing an IdToken)	
Purpose	To verify if the Charging Station is able to store the identifiers correctly upon an AutorizeResponse according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	- AuthCacheCtrlr.AuthCacheAvailable is implemented with value true - The Charging Station supports authorization methods other than NoAuthorization	
Before (Preparations)	**Configuration State:** **AuthCacheEnabled** is *true* (If implemented) **LocalPreAuthorize** is *true* **LocalAuthListEnabled** is *true*	
	Memory State: N/a	
	Reusable State(s): **State is** *EVConnectedPreSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Present valid idToken*	
	1. The Charging Station sends an **AuthorizeRequest**	**2.** The OCTT responds with an **AuthorizeResponse** with **idTokenInfo.status** *Accepted*
	3. The Charging Station sends a **TransactionEventRequest** Note(s): *- This step needs to be executed when **TxStartPoint** contains ParkingBayOccupancy, EVConnected, Authorized, or PowerPathClosed*	**4.** The OCTT responds with a **TransactionEventResponse** with **idTokenInfo.status** *Accepted*
	5. Execute **Reusable State** *EnergyTransferStarted*	
	6. Execute **Reusable State** *EVConnectedPostSession*	
	7. Execute **Reusable State** *EVDisconnected*	
	8. Execute **Reusable State** *ParkingBayUnoccupied*	
	9. Execute **Reusable State** *ParkingBayOccupied*	
	10. Execute **Reusable State** *EVConnectedPreSession*	
	Manual Action: *Present same valid idToken*	
	12. The Charging Station sends a **TransactionEventRequest** Note(s): *- This step needs to be executed when **TxStartPoint** contains Authorized OR the transaction already started. So in the case **TxStartPoint** contains ParkingBayOccupancy.*	**13.** The OCTT responds with a **TransactionEventResponse** with **idTokenInfo.status** *Invalid*

Test case name	**Store Authorization Data in the Authorization Cache - Update on AuthorizeResponse**
Tool validations	* Step 1: Message **AuthorizeRequest** - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>* * Step 3: Message **TransactionEventRequest** - **triggerReason** *Authorized* - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>* if transaction was already started - **eventType** *Updated* else - **eventType** *Started* * Step 12: Message **TransactionEventRequest** - **triggerReason** *Authorized* - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>* if transaction was already started - **eventType** *Updated* else - **eventType** *Started*
	Post scenario validations: - N/a

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

Table 90. Test Case Id: TC_C_34_CS

Test case name	**Store Authorization Data in the Authorization Cache - Update on TransactionResponse**	
Test case Id	TC_C_34_CS	
Use case Id(s)	C10	
Requirement(s)	C10_FR_05, C12.FR.06	
System under test	Charging Station	
Description	This test case covers how the Charging Station autonomously stores a record of previously presented identifiers that have been successfully authorized by the CSMS in the Authorization Cache. (Successfully meaning: a response received on a message containing an IdToken)	
Purpose	To verify if the Charging Station is able to store the identifiers correctly upon an TransactionResponse according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	- AuthCacheCtrlr.AuthCacheAvailable is implemented with value true - The Charging Station supports authorization methods other than NoAuthorization	
Before (Preparations)	**Configuration State:** **AuthCacheEnabled** is *true* (If implemented) **LocalPreAuthorize** is *true* **LocalAuthListEnabled** is *true*	
	Memory State: *IdTokenCached* for <Configured valid IdToken fields>	
	Reusable State(s): **State is** *EVConnectedPreSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Present valid idToken*	
	1. The Charging Station sends a **TransactionEventRequest**	**2.** The OCTT responds with a **TransactionEventResponse** with **idTokenInfo.status** *Invalid*
	3. Execute **Reusable State** *EVDisconnected*	
	4. Execute **Reusable State** *ParkingBayUnoccupied*	
	5. Execute **Reusable State** *ParkingBayOccupied*	
	6. Execute **Reusable State** *EVConnectedPreSession*	
	Manual Action: *Present same valid idToken*	
	7. The Charging Station sends an **AuthorizeRequest**	**8.** The OCTT responds with an **AuthorizeResponse** with **idTokenInfo.status** *Invalid*
Tool validations	* Step 1: Message **TransactionEventRequest** - **triggerReason** *Authorized* - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>* if transaction was already started - **eventType** *Updated* else - **eventType** *Started* * Step 7: Message **AuthorizeRequest** - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>*	
	Post scenario validations: - N/a	

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

Table 91. Test Case Id: TC_C_36_CS

Test case name	**Store Authorization Data in the Authorization Cache - AuthCacheCtrlr.LocalPreAuthorize = false**	
Test case Id	TC_C_36_CS	
Use case Id(s)	C10	
Requirement(s)	C10_FR_11	
System under test	Charging Station	
Description	This test case covers how the Charging Station autonomously stores a record of previously presented identifiers that have been successfully authorized by the CSMS in the Authorization Cache. (Successfully meaning: a response received on a message containing an IdToken)	
Purpose	To verify if the Charging Station is able to ignore the Authorization Cache feature when LocalPreAuthorize is set to false according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	- AuthCacheCtrlr.AuthCacheAvailable is implemented - The Charging Station supports authorization methods other than NoAuthorization	
Before (Preparations)	**Configuration State:** **AuthCacheEnabled** is *true* **LocalPreAuthorize** is *false*	
	Memory State: *IdTokenCached* for <Configured valid IdToken fields>	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Present valid idToken which is configured in the Authorization Cache*	
	1. The Charging Station sends an **AuthorizeRequest**	**2.** The OCTT responds with an **AuthorizeResponse** with **idTokenInfo.status** *Invalid*
Tool validations	* Step 1: Message **AuthorizeRequest** - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>*	
	Post scenario validations: - N/a	

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

Table 92. Test Case Id: TC_C_37_CS

Test case name	**Clear Authorization Data in Authorization Cache - Accepted**	
Test case Id	TC_C_37_CS	
Use case Id(s)	C11	
Requirement(s)	C11_FR_01, C11.FR.02, C11.FR.03	
System under test	Charging Station	
Description	This test case covers how the Charging Station autonomously stores a record of previously presented identifiers that have been successfully authorized by the CSMS in the Authorization Cache. (Successfully meaning: a response received on a message containing an IdToken)	
Purpose	To verify if the Charging Station is able to clear all identifiers from the Authorization Cache according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	- AuthCacheCtrlr.AuthCacheAvailable is implemented with value true - The Charging Station supports authorization methods other than NoAuthorization	
Before (Preparations)	**Configuration State:** **AuthCacheEnabled** is *true* (If implemented)	
	Memory State: *IdTokenCached* for <Configured valid IdToken fields>	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **ClearCacheResponse**	**1.** The OCTT sends a **ClearCacheRequest**
	3. Execute **Reusable State** *ParkingBayOccupied*	
	4. Execute **Reusable State** *EVConnectedPreSession*	
	Manual Action: *Present valid idToken which was configured in the Authorization Cache*	
	5. The Charging Station sends an **AuthorizeRequest**	**6.** The OCTT responds with an **AuthorizeResponse** with **idTokenInfo.status** *Accepted*
	7. The Charging Station sends an **TransactionEventRequest** with **triggerReason** *Authorized*	**8.** The OCTT responds with an **TransactionEventResponse** with
	9. Execute **Reusable State** *EnergyTransferStarted*	
Tool validations	* Step 2: Message **ClearCacheResponse** - **status** *Accepted* * Step 5: Message **AuthorizeRequest** - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>*	
	Post scenario validations: - N/a	

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

Table 93. Test Case Id: TC_C_38_CS

Test case name	**Clear Authorization Data in Authorization Cache - Rejected**	
Test case Id	TC_C_38_CS	
Use case Id(s)	C11	
Requirement(s)	C11_FR_01, C11.FR.02, C11.FR.04	
System under test	Charging Station	
Description	This test case covers how the Charging Station autonomously stores a record of previously presented identifiers that have been successfully authorized by the CSMS in the Authorization Cache. (Successfully meaning: a response received on a message containing an IdToken)	
Purpose	To verify if the Charging Station is able to correctly respond on a request from the CSMS to clear all identifiers from the Authorization Cache while the feature is disabled according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	- AuthCacheCtrlr.AuthCacheAvailable is implemented with value true - **AuthCacheCtrlr.LocalPreAuthorize** is implemented - The Charging Station supports authorization methods other than NoAuthorization	
Before (Preparations)	**Configuration State:** **AuthCacheEnabled** is *false* (If implemented)	
	Memory State: *IdTokenCached* for <Configured valid IdToken fields>	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **ClearCacheResponse**	**1.** The OCTT sends a **ClearCacheRequest**
Tool validations	* Step 2: Message **ClearCacheResponse** - **status** *Rejected*	
	Post scenario validations: - N/a	

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

Table 94. Test Case Id: TC_C_39_CS

Test case name	**Authorization by GroupId - Success**	
Test case Id	TC_C_39_CS	
Use case Id(s)	C09	
Requirement(s)	C09_FR_02, C09_FR_03, C09_FR_05	
System under test	Charging Station	
Description	This test case covers how a Charging Station can authorize an action for an EV Driver based on GroupId information. This could for example be used if 2 people regularly use the same EV: they can use their own IdToken (e.g. RFID card), and can deauthorize transactions that were started with the other idToken (with the same GroupId).	
Purpose	To verify if the Charging Station is able to correctly handle the Authorization of idTokens with the same GroupId according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	- The Charging Station supports authorization methods other than NoAuthorization	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): **State is** *EVConnectedPreSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Present valid idToken with <Configured GroupId>*	
	1. The Charging Station sends an **AuthorizeRequest**	**2.** The OCTT responds with an **AuthorizeResponse** with - **idTokenInfo.status** *Accepted* - **idTokenInfo.groupIdToken.idToken** *<Configured groupIdToken>*
	3. The Charging Station sends a **TransactionEventRequest** Note(s): *- This step needs to be executed when **TxStartPoint** contains Authorized OR the transaction already started. So in the case **TxStartPoint** contains ParkingBayOccupancy*	**4.** The OCTT responds with a **TransactionEventResponse** with - **idTokenInfo.status** *Accepted* - **idTokenInfo.groupIdToken.idToken** *<Configured groupIdToken>*
	5. Execute **Reusable State** *EnergyTransferStarted*	
	Manual Action: *Present other valid idToken with <Configured GroupId>*	
	6. The Charging Station sends an **AuthorizeRequest**	**7.** The OCTT responds with an **AuthorizeResponse** with - **idTokenInfo.status** *Accepted* - **idTokenInfo.groupIdToken.idToken** *<Configured groupIdToken>*
	8. The Charging Station sends a **TransactionEventRequest**	**9.** The OCTT responds with a **TransactionEventResponse** with - **idTokenInfo.status** *Accepted* - **idTokenInfo.groupIdToken.idToken** *<Configured groupIdToken>*
	10. Execute **Reusable State** *EVConnectedPostSession*	
	11. Execute **Reusable State** *EVDisconnected*	
	12. Execute **Reusable State** *ParkingBayUnoccupied*	

Test case name	**Authorization by GroupId - Success**
Tool validations	* Step 1: Message **AuthorizeRequest** - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>* * Step 3: Message **TransactionEventRequest** - **triggerReason** *Authorized* - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>* if transaction was already started - **eventType** *Updated* else - **eventType** *Started* * Step 6: Message **AuthorizeRequest** - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>* * Step 8: Message **TransactionEventRequest** - **triggerReason** *StopAuthorized* - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>*
	Post scenario validations: - N/a

Table 95. Test Case Id: TC_C_40_CS

Test case name	**Authorization by GroupId - Success with Local Authorization List**	
Test case Id	TC_C_40_CS	
Use case Id(s)	C09	
Requirement(s)	C09_FR_02, C09_FR_03, C09_FR_07	
System under test	Charging Station	
Description	This test case covers how a Charging Station can authorize an action for an EV Driver based on GroupId information. This could for example be used if 2 people regularly use the same EV: they can use their own IdToken (e.g. RFID card), and can deauthorize transactions that were started with the other idToken (with the same GroupId).	
Purpose	To verify if the Charging Station is able to correctly handle the Authorization of idTokens with the same GroupId when stored in the Local Authorization List according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	- LocalAuthListCtrlr.LocalAuthListAvailable is implemented with value true - The Charging Station supports authorization methods other than NoAuthorization	
Before (Preparations)	**Configuration State:** **LocalAuthListEnabled** is *true* (If implemented) **LocalPreAuthorize** is *true* (If implemented)	
	Memory State: Two known valid idTokens are configured in the Local Authorization List with the same GroupId	
	Reusable State(s): **State is** *EVConnectedPreSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Present valid idToken with <Configured GroupId> which is configured in the Local Authorization List*	
	1. The Charging Station sends a **TransactionEventRequest** Note(s): *- This step needs to be executed when **TxStartPoint** contains Authorized OR the transaction already started. So in the case **TxStartPoint** contains ParkingBayOccupancy*	**2.** The OCTT responds with a **TransactionEventResponse** with - **idTokenInfo.status** *Accepted* - **idTokenInfo.groupIdToken.idToken** *<Configured groupIdToken>*
	3. Execute **Reusable State** *EnergyTransferStarted*	
	Manual Action: *Present valid idToken2 with <Configured GroupId> which is configured in the Authorization Cache*	
	4. Execute **Reusable State** *StopAuthorized*	
	5. Execute **Reusable State** *EVConnectedPostSession*	
	6. Execute **Reusable State** *EVDisconnected*	
	7. Execute **Reusable State** *ParkingBayUnoccupied*	

Test case name	**Authorization by GroupId - Success with Local Authorization List**
Tool validations	* Step 1: Message **TransactionEventRequest** - **triggerReason** *Authorized* - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>* if transaction was already started - **eventType** *Updated* else - **eventType** *Started* * Step 4: Message **TransactionEventRequest** - **triggerReason** *StopAuthorized* - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>* - **eventType** *Updated*
	Post scenario validations: - N/a

Table 96. Test Case Id: TC_C_41_CS

Test case name	**Authorization by GroupId - Success with Authorization Cache**	
Test case Id	TC_C_41_CS	
Use case Id(s)	C09	
Requirement(s)	C09_FR_02, C09_FR_03, C09_FR_07	
System under test	Charging Station	
Description	This test case covers how a Charging Station can authorize an action for an EV Driver based on GroupId information. This could for example be used if 2 people regularly use the same EV: they can use their own IdToken (e.g. RFID card), and can deauthorize transactions that were started with the other idToken (with the same GroupId).	
Purpose	To verify if the Charging Station is able to correctly handle the Authorization of idTokens with the same GroupId when stored in the Authorization Cache according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	- AuthCacheCtrlr.AuthCacheAvailable is implemented with value true - The Charging Station supports authorization methods other than NoAuthorization	
Before (Preparations)	**Configuration State:** **AuthCacheEnabled** is *true* (If implemented) **LocalPreAuthorize** is *true* (If implemented)	
	Memory State: *IdTokenCached* for <Configured valid IdToken fields> *IdTokenCached* for <Configured valid IdToken2 fields>	
	Reusable State(s): **State is** *EVConnectedPreSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Present valid idToken with <Configured GroupId> which is configured in the Authorization Cache*	
	1. The Charging Station sends a **TransactionEventRequest** Note(s): *- This step needs to be executed when **TxStartPoint** contains Authorized OR the transaction already started. So in the case **TxStartPoint** contains ParkingBayOccupancy*	**2.** The OCTT responds with a **TransactionEventResponse** with - **idTokenInfo.status** *Accepted* - **idTokenInfo.groupIdToken.idToken** *<Configured groupIdToken>*
	3. Execute **Reusable State** *EnergyTransferStarted*	
	Manual Action: *Present valid idToken2 with <Configured GroupId> which is configured in the Authorization Cache*	
	4. Execute **Reusable State** *StopAuthorized*	
	5. Execute **Reusable State** *EVConnectedPostSession*	
	6. Execute **Reusable State** *EVDisconnected*	
	7. Execute **Reusable State** *ParkingBayUnoccupied*	

Test case name	**Authorization by GroupId - Success with Authorization Cache**
Tool validations	* Step 1: Message **TransactionEventRequest** - **triggerReason** *Authorized* - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>* if transaction was already started - **eventType** *Updated* else - **eventType** *Started* * Step 4: Message **TransactionEventRequest** - **triggerReason** *StopAuthorized* - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>*
	Post scenario validations: - N/a

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

Table 97. Test Case Id: TC_C_42_CS

Test case name	**Authorization by GroupId - Not stopped by GroupId**	
Test case Id	TC_C_42_CS	
Use case Id(s)	C09	
Requirement(s)	C09_FR_11	
System under test	Charging Station	
Description	This test case covers how a Charging Station can authorize an action for an EV Driver based on GroupId information. This could for example be used if 2 people regularly use the same EV: they can use their own IdToken (e.g. RFID card), and can deauthorize transactions that were started with the other idToken (with the same GroupId).	
Purpose	To verify if the Charging Station is able to correctly handle the Authorization of idTokens with the same GroupId, while one of them is invalid, according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	- The Charging Station supports authorization methods other than NoAuthorization	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): **State is** *EVConnectedPreSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Present valid idToken with <Configured GroupId>*	
	1. The Charging Station sends an **AuthorizeRequest**	**2.** The OCTT responds with an **AuthorizeResponse** with - **idTokenInfo.status** *Accepted* - **idTokenInfo.groupIdToken.idToken** <*Configured groupIdToken*>
	3. The Charging Station sends a **TransactionEventRequest** Note(s): *- This step needs to be executed when **TxStartPoint** contains Authorized OR the transaction already started. So in the case **TxStartPoint** contains ParkingBayOccupancy*	**4.** The OCTT responds with a **TransactionEventResponse** with - **idTokenInfo.status** *Accepted* - **idTokenInfo.groupIdToken.idToken** <*Configured groupIdToken*>
	5. Execute **Reusable State** *EnergyTransferStarted*	
	Manual Action: *Present invalid idToken with <Configured GroupId>*	
	6. The Charging Station sends an **AuthorizeRequest**	**7.** The OCTT responds with an **AuthorizeResponse** with - **idTokenInfo.status** *Invalid* - **idTokenInfo.groupIdToken.idToken** <*Configured groupIdToken*>
	Note(s): *OCTT will wait to see if CS indeed doesn't send a TransactionEventRequest*	

Test case name	**Authorization by GroupId - Not stopped by GroupId**
Tool validations	* Step 1: Message **AuthorizeRequest** - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>* * Step 3: Message **TransactionEventRequest** - **triggerReason** *Authorized* - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>* if transaction was already started - **eventType** *Updated* else - **eventType** *Started* * Step 6: Message **AuthorizeRequest** - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>*
	Post scenario validations: - The energy transfer is not stopped

Table 98. Test Case Id: TC_C_43_CS

Test case name	**Authorization by GroupId - Invalid status with Local Authorization List**	
Test case Id	TC_C_43_CS	
Use case Id(s)	C09	
Requirement(s)	C09_FR_02, C09_FR_03, C09_FR_07	
System under test	Charging Station	
Description	This test case covers how a Charging Station can authorize an action for an EV Driver based on GroupId information. This could for example be used if 2 people regularly use the same EV: they can use their own IdToken (e.g. RFID card), and can deauthorize transactions that were started with the other idToken (with the same GroupId).	
Purpose	To verify if the Charging Station is able to correctly handle the Authorization of idTokens with the same GroupId when stored in the Local Authorization List, but one of them is invalid, according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	- LocalAuthListCtrlr.LocalAuthListAvailable is implemented with value true - The Charging Station supports authorization methods other than NoAuthorization	
Before (Preparations)	**Configuration State:** **LocalAuthListEnabled** is *true* (If implemented) **LocalPreAuthorize** is *true* (If implemented)	
	Memory State: Two known idTokens are configured in the Local Authorization List with the same GroupId, one is valid and one is invalid.	
	Reusable State(s): **State is** *EVConnectedPreSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Present valid idToken with <Configured GroupId> which is configured in the Local Authorization List*	
	1. The Charging Station sends a **TransactionEventRequest** Note(s): *- This step needs to be executed when **TxStartPoint** contains Authorized OR the transaction already started. So in the case **TxStartPoint** contains ParkingBayOccupancy*	**2.** The OCTT responds with a **TransactionEventResponse** with - **idTokenInfo.status** *Accepted* - **idTokenInfo.groupIdToken.idToken** *<Configured groupIdToken>*
	3. Execute **Reusable State** *EnergyTransferStarted*	
	Manual Action: *Present invalid idToken with <Configured GroupId> which is configured in the Local Authorization List*	
	4. The Charging Station sends an **AuthorizeRequest**	**5.** The OCTT responds with an **AuthorizeResponse** with - **idTokenInfo.status** *Invalid* - **idTokenInfo.groupIdToken.idToken** *<Configured groupIdToken>*
	Note(s): *OCTT will wait to see if CS indeed doesn't send a TransactionEventRequest*	

Test case name	**Authorization by GroupId - Invalid status with Local Authorization List**
Tool validations	* Step 1: Message **TransactionEventRequest** - **triggerReason** *Authorized* - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>* if transaction was already started - **eventType** *Updated* else - **eventType** *Started* * Step 4: Message **AuthorizeRequest** - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>* * Step 6: Message **TransactionEventRequest** - **triggerReason** *StopAuthorized* - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>* - **eventType** *Updated*
	Post scenario validations: - N/a

Table 99. Test Case Id: TC_C_44_CS

Test case name	**Authorization by GroupId - Invalid status with Authorization Cache**	
Test case Id	TC_C_44_CS	
Use case Id(s)	C09	
Requirement(s)	C09_FR_02, C09_FR_03, C09_FR_07	
System under test	Charging Station	
Description	This test case covers how a Charging Station can authorize an action for an EV Driver based on GroupId information. This could for example be used if 2 people regularly use the same EV: they can use their own IdToken (e.g. RFID card), and can deauthorize transactions that were started with the other idToken (with the same GroupId).	
Purpose	To verify if the Charging Station is able to correctly handle the Authorization of idTokens with the same GroupId when stored in the Authorization Cache, but one of them is invalid, according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	- AuthCacheCtrlr.AuthCacheAvailable is implemented with value true - The Charging Station supports authorization methods other than NoAuthorization	
Before (Preparations)	**Configuration State:** **AuthCacheEnabled** is *true* (If implemented) **LocalPreAuthorize** is *true* (If implemented) **AuthCacheCtrlrDisablePostAuthorize** is *false* (If implemented)	
	Memory State: *IdTokenCached* for <Configured valid IdToken fields> *IdTokenCached* for <Configured invalid IdToken fields>	
	Reusable State(s): **State is** *EVConnectedPreSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Present valid idToken with <Configured GroupId> which is configured in the Authorization Cache*	
	1. The Charging Station sends a **TransactionEventRequest** Note(s): *- This step needs to be executed when **TxStartPoint** contains Authorized OR the transaction already started. So in the case **TxStartPoint** contains ParkingBayOccupancy*	**2.** The OCTT responds with a **TransactionEventResponse** with - **idTokenInfo.status** *Accepted* - **idTokenInfo.groupIdToken.idToken** *<Configured groupIdToken>*
	3. Execute **Reusable State** *EnergyTransferStarted*	
	Manual Action: *Present invalid idToken with <Configured GroupId> which is configured in the Authorization Cache*	
	4. The Charging Station sends an **AuthorizeRequest**	**5.** The OCTT responds with an **AuthorizeResponse** with - **idTokenInfo.status** *Invalid* - **idTokenInfo.groupIdToken.idToken** *<Configured groupIdToken>*
	Note(s): *OCTT will wait to see if CS indeed doesn't send a TransactionEventRequest*	

Test case name	**Authorization by GroupId - Invalid status with Authorization Cache**
Tool validations	* Step 1: Message **TransactionEventRequest** - **triggerReason** *Authorized* - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>* if transaction was already started - **eventType** *Updated* else - **eventType** *Started* * Step 4: Message **AuthorizeRequest** - **idToken.idToken** *<Configured invalid_idtoken_idtoken>* - **idToken.type** *<Configured invalid_idtoken_type>*
	Post scenario validations: - N/a

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

Table 100. Test Case Id: TC_C_45_CS

Test case name	**Authorization by GroupId - Master pass - Not able to start transaction + groupId**	
Test case Id	TC_C_45_CS	
Use case Id(s)	C09	
Requirement(s)	C16.FR.03	
System under test	Charging Station	
Description	This test case covers how a Charging Station can authorize an action for an EV Driver based on GroupId information. This could for example be used if 2 people regularly use the same EV: they can use their own IdToken (e.g. RFID card), and can deauthorize transactions that were started with the other idToken (with the same GroupId).	
Purpose	To verify if the Charging Station is able to correctly handle the Authorization of an idToken with the same GroupId as the MasterPassGroupId according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	- The Charging station supports MasterPass feature. - The Charging Station supports authorization methods other than NoAuthorization	
Before (Preparations)	**Configuration State:** **TxCtrlr.TxStartPoint** should contain *Authorized* or *PowerPathClosed* and not contain *ParkingBayOccupancy* or *EVConnected* **AuthCtrlr.MasterPassGroupId** is *<Configured MasterPassGroupId>*	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Present configured masterpass idToken*	
	1. The Charging Station sends an **AuthorizeRequest**	**2.** The OCTT responds with an **AuthorizeResponse** with - **idTokenInfo.status** *Accepted* - **idTokenInfo.groupIdToken.idToken** *<Configured masterPassGroupId>*
	Note: *The Charging Station will not authorize the transaction and send a TransactionEventRequest (in case of TxStartPoint Authorized).*	
	3. Execute **Reusable State** *EVConnectedPreSession*	
	4. The Charging Station will NOT send a **TransactionEventRequest** with **chargingState** *Charging* and **triggerReason** *ChargingStateChanged*	
Tool validations	* Step 1: Message **AuthorizeRequest** - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>*	
	Post scenario validations: - N/a	

Table 101. Test Case Id: TC_C_46_CS

Test case name	**Store Authorization Data in the Authorization Cache - AuthCacheLifeTime**	
Test case Id	TC_C_46_CS	
Use case Id(s)	C10	
Requirement(s)	C10_FR_08	
System under test	Charging Station	
Description	This test case covers how the Charging Station autonomously stores a record of previously presented identifiers that have been successfully authorized by the CSMS in the Authorization Cache. (Successfully meaning: a response received on a message containing an IdToken)	
Purpose	To verify if the Charging Station is able to correctly remove an idToken when this one is not reused again within the specified amount of time (AuthCacheLifeTime) according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	- AuthCacheCtrlr.AuthCacheAvailable is implemented with value true - Configuration variable AuthCacheLifeTime is implemented - The Charging Station supports authorization methods other than NoAuthorization	
Before (Preparations)	**Configuration State:** **AuthCacheLifeTime** is *<Configured TransactionDuration>* **AuthCacheCtrlr.LocalPreAuthorize** is *true* (If implemented)	
	Memory State: *IdTokenCached* *<Configured valid idtoken fields>*	
	Reusable State(s): **State is** *EVConnectedPreSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. *Wait for <Configured Transaction Duration> seconds*	
	2. Execute **Reusable State** *Authorized* (local)	
Tool validations	N/a	
	Post scenario validations: - N/a	

Table 102. Test Case Id: TC_C_47_CS

Test case name	**Stop Transaction with a Master Pass - With UI - All transactions**	
Test case Id	TC_C_47_CS	
Use case Id(s)	C16	
Requirement(s)	C16_FR_01	
System under test	Charging Station	
Description	This test case covers how somebody with a Master Pass (User) can stop (selected) ongoing transactions, so the cable becomes unlocked. This Master Pass can be configured in: MasterPassGroupId. This could for example be usefull for Law Enforcement officials.	
Purpose	To verify if the Charging Station is able to correctly stop all transactions when an idToken which has the MasterPass as GroupId is used and the user has selected to stop all transactions in the User Interface according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	- The Charging Station supports at least one authorization method described at the following Use cases; C01, C04. - Charging station has a User Interface - The Charging Station supports authorization methods other than NoAuthorization	
Before (Preparations)	**Configuration State:** AuthCtrlr.MastersPassGroupId is configured	
	Memory State: N/a	
	Reusable State(s): **State is** *EnergyTransferStarted* for all EVSE	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Present configured masterpass idToken*	
	1. The Charging Station sends an **AuthorizeRequest**	**2.** The OCTT responds with an **AuthorizeResponse** with **idTokenInfo.status** *Accepted* **idTokenInfo.groupIdToken.idToken** *<Configured masterPassGroupId>*
	Manual Action: *Select to stop all transactions*	
	3. The Charging Station sends a **TransactionEventRequest** for all EVSE	**4.** The OCTT responds with a **TransactionEventResponse** with **idTokenInfo.status** *Accepted* **idTokenInfo.groupIdToken.idToken** *<Configured masterPassGroupId>* for all EVSE
	5. Execute **Reusable State** *EVConnectedPostSession* for all EVSE	
	6. Execute **Reusable State** *EVDisconnected* for all EVSE	
	7. Execute **Reusable State** *ParkingBayUnoccupied* for all EVSE	
Tool validations	* Step 1: Message **AuthorizeRequest** - **idToken.idToken** *<Configured masterpass_idtoken_idtoken>* - **idToken.type** *<Configured masterpass_idtoken_type>* * Step 3: Message **TransactionEventRequest** - **transactionInfo.stoppedReason** *MasterPass* - **idToken** *omit* or - **idToken.idToken** *<Configured masterpass_idtoken_idtoken>* and - **idToken.type** *<Configured masterpass_idtoken_type>*	
	Post scenario validations: - N/a	

Table 103. Test Case Id: TC_C_48_CS

Test case name	**Stop Transaction with a Master Pass - With UI - With UI - Specific transactions**	
Test case Id	TC_C_48_CS	
Use case Id(s)	C16	
Requirement(s)	C16_FR_01	
System under test	Charging Station	
Description	This test case covers how somebody with a Master Pass (User) can stop (selected) ongoing transactions, so the cable becomes unlocked. This Master Pass can be configured in: MasterPassGroupId. This could for example be usefull for Law Enforcement officials.	
Purpose	To verify if the Charging Station is able to correctly stop a transaction when an idToken which has the MasterPass as GroupId is used and the user has selected to stop one transaction in the User Interface according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	- The Charging Station supports at least one authorization method described at the following Use cases; C01, C04. - Charging station has a User Interface - The Charging Station supports authorization methods other than NoAuthorization	
Before (Preparations)	**Configuration State:** AuthCtrlr.MastersPassGroupId is configured	
	Memory State: N/a	
	Reusable State(s): **State is** *EnergyTransferStarted* for all EVSE	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Present configured masterpass idToken*	
	1. The Charging Station sends an **AuthorizeRequest**	**2.** The OCTT responds with an **AuthorizeResponse** with **idTokenInfo.status** *Accepted* **idTokenInfo.groupIdToken.idToken** *<Configured masterPassGroupId>*
	Manual Action: *Select to stop the transaction on EVSE 1*	
	3. The Charging Station sends a **TransactionEventRequest**	**4.** The OCTT responds with a **TransactionEventResponse** with **idTokenInfo.status** *Accepted* **idTokenInfo.groupIdToken.idToken** *<Configured masterPassGroupId>*
	5. Execute **Reusable State** *EVConnectedPostSession*	
	6. Execute **Reusable State** *EVDisconnected*	
	7. Execute **Reusable State** *ParkingBayUnoccupied*	
Tool validations	* Step 1: Message **AuthorizeRequest** - **idToken.idToken** *<Configured masterpass_idtoken_idtoken>* - **idToken.type** *<Configured masterpass_idtoken_type>* * Step 3: Message **TransactionEventRequest** - **transactionInfo.stoppedReason** *MasterPass* - **idToken** *omit* or - **idToken.idToken** *<Configured masterpass_idtoken_idtoken>* and - **idToken.type** *<Configured masterpass_idtoken_type>*	
	Post scenario validations: - All other EVSE still transfer energy	

Table 104. Test Case Id: TC_C_49_CS

Test case name	**Stop Transaction with a Master Pass - Without UI**	
Test case Id	TC_C_49_CS	
Use case Id(s)	C16	
Requirement(s)	C16_FR_02	
System under test	Charging Station	
Description	This test case covers how somebody with a Master Pass (User) can stop (selected) ongoing transactions, so the cable becomes unlocked. This Master Pass can be configured in: MasterPassGroupId. This could for example be usefull for Law Enforcement officials.	
Purpose	To verify if the Charging Station is able to correctly stop all transactions when an idToken which has the MasterPass as GroupId is used and the Charging station does not have an User Interface according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	- The Charging Station supports at least one authorization method described at the following Use cases; C01, C04. - The Charging Station supports authorization methods other than NoAuthorization	
Before (Preparations)	**Configuration State:** AuthCtrlr.MastersPassGroupId is configured	
	Memory State: N/a	
	Reusable State(s): **State is** *EnergyTransferStarted* for EVSEId *1* and EVSEId *2* if the Charging Station has more than one EVSE. With: - *<Configured valid_idtoken>* for EVSE 1 - *<Configured valid_idtoken2>* for EVSE 2	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Present configured masterpass idToken*	
	1. The Charging Station sends an **AuthorizeRequest**	**2.** The OCTT responds with an **AuthorizeResponse** with **idTokenInfo.status** *Accepted* **idTokenInfo.groupIdToken.idToken** *<Configured masterPassGroupId>*
	3. The Charging Station sends a **TransactionEventRequest** for EVSE 1 (and 2)	**4.** The OCTT responds with a **TransactionEventResponse** with **idTokenInfo.status** *Accepted* **idTokenInfo.groupIdToken.idToken** *<Configured masterPassGroupId>* for EVSE 1 (and 2)
	5. Execute **Reusable State** *EVConnectedPostSession* for EVSE 1 (and 2)	
	6. Execute **Reusable State** *EVDisconnected* for EVSE 1 (and 2)	
	7. Execute **Reusable State** *ParkingBayUnoccupied* for EVSE 1 (and 2)	
Tool validations	* Step 1: Message **AuthorizeRequest** - **idToken.idToken** *<Configured masterpass_idtoken_idtoken>* - **idToken.type** *<Configured masterpass_idtoken_type>* * Step 3: Message **TransactionEventRequest** - **transactionInfo.stoppedReason** *MasterPass* - **idToken** *omit* or - **idToken.idToken** *<Configured masterpass_idtoken_idtoken>* and - **idToken.type** *<Configured masterpass_idtoken_type>*	
	Post scenario validations: - N/a	

Table 105. Test Case Id: TC_C_21_CS

Test case name	**Offline authorization through local authorization list - Accepted**	
Test case Id	TC_C_21_CS	
Use case Id(s)	C13	
Requirement(s)	C13.FR.02	
System under test	Charging Station	
Description	This test case describes how to authorize an IdToken via the Local Authorization List while the Charging Station is online. When offline the Charging Station can then locally authorize the IdToken.	
Purpose	To verify if the Charging Station is able to authorize an idToken which has status "Accepted" in its local authorization list according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	- LocalAuthListCtrlr.LocalAuthListAvailable is implemented with value true - The Charging Station supports authorization methods other than NoAuthorization	
Before (Preparations)	**Configuration State:** **LocalAuthListEnabled** is *true* (If implemented) **LocalPreAuthorize** is *true* (If implemented) **OfflineTxForUnknownIdEnabled** is *false* (If implemented) **LocalAuthorizeOffline** is *true*	
	Memory State: *IdTokenLocalAuthList* for *<Configured valid idtoken fields>*	
	Reusable State(s): **State is** *StartOfflineTransaction*	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Present idToken.*	
	Manual Action: *Unplug cable.*	
	Manual Action: *Drive out of parkingbay.*	
	1. The Charging Stations sends a **TransactionEventRequest** Note(s): *- The Charging Station will empty its Transaction message queue. This will contain one or more TransactionEventRequest messages*	**2.** The OCTT responds with a **TransactionEventResponse** Note(s): *- The OCTT will respond to the TransactionEventRequest containing the idToken, with* ***idtokenInfo.status*** *Accepted*
	3. Execute **Reusable State** *ParkingBayUnoccupied*	
Tool validations	* Step 1: Message(s) before the StopAuthorize: **TransactionEventRequests** - **offline** must be *true* One of the Message(s): **TransactionEventRequest** - **TriggerReason** must be *StopAuthorized*	
	Post scenario validations: N/a	

Table 106. Test Case Id: TC_C_22_CS

Test case name	**Offline authorization through local authorization list - Invalid**	
Test case Id	TC_C_22_CS	
Use case Id(s)	C13	
Requirement(s)	C13.FR.02	
System under test	Charging Station	
Description	This test case describes how to authorize an IdToken via the Local Authorization List while the Charging Station is online. When offline the Charging Station can then locally authorize the IdToken.	
Purpose	To verify if the Charging Station is able to respond correctly to an idToken which has status "Invalid" in its local authorization list according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	- LocalAuthListCtrlr.LocalAuthListAvailable is implemented with value true - The Charging Station supports authorization methods other than NoAuthorization	
Before (Preparations)	**Configuration State:** **LocalAuthListEnabled** is *true* (If implemented) **LocalPreAuthorize** is *true* (If implemented) **OfflineTxForUnknownIdEnabled** is *false* (If implemented) **LocalAuthorizeOffline** is *true* **OfflineThreshold** is *<Configured RetryBackOffWaitMinimum_duration> + 60.0* **RetryBackOffWaitMinimum** is *<Configured RetryBackOffWaitMinimum_duration>* **RetryBackOffRandomRange** is *0*	
	Memory State: *IdTokenLocalAuthList* for *<Configured invalid idtoken fields>*	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. *The OCTT closes the WebSocket connection AND does not accept a reconnect.*	
	2. Manual Action: *Drive EV into parking bay.*	
	3. Manual Action: *Present idToken.*	
	4. *The OCTT accepts reconnection attempt from the Charging Station.*	
	5. The Charging Stations sends a **TransactionEventRequest** Note(s): - *The Charging Station will empty its Transaction message queue, this will only contain TransactionEventRequests if TxStartPoint was ParkingBayOccupancy and a Transaction was started.*	**6.** The OCTT responds with a **TransactionEventResponse**
Tool validations	* Step 5: Message **TransactionEventRequest** - **offline** must be *true* - **TriggerReason** must be *EVDetected*	
	Post scenario validations: N/a	

Table 107. Test Case Id: TC_C_23_CS

Test case name	**Offline authorization through local authorization list - Blocked**	
Test case Id	TC_C_23_CS	
Use case Id(s)	C13	
Requirement(s)	C13.FR.02	
System under test	Charging Station	
Description	This test case describes how to authorize an IdToken via the Local Authorization List while the Charging Station is online. When offline the Charging Station can then locally authorize the IdToken.	
Purpose	To verify if the Charging Station is able to respond correctly to an idToken which has status "Blocked" in its local authorization list according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	- LocalAuthListCtrlr.LocalAuthListAvailable is implemented with value true - The Charging Station supports authorization methods other than NoAuthorization	
Before (Preparations)	**Configuration State:** **LocalAuthListEnabled** is *true* (If implemented) **LocalPreAuthorize** is *true* (If implemented) **OfflineTxForUnknownIdEnabled** is *false* (If implemented) **LocalAuthorizeOffline** is *true* **OfflineThreshold** is *<Configured RetryBackOffWaitMinimum_duration> + 60.0* **RetryBackOffWaitMinimum** is *<Configured RetryBackOffWaitMinimum_duration>* **RetryBackOffRandomRange** is *0*	
	Memory State: *IdTokenLocalAuthList* for *<Configured blocked idtoken fields>*	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. *The OCTT closes the WebSocket connection AND does not accept a reconnect.*	
	2. Manual Action: *Drive EV into parking bay.*	
	3. Manual Action: *Present idToken.*	
	4. *The OCTT accepts reconnection attempt from the Charging Station.*	
	5. The Charging Stations sends a **TransactionEventRequest** Note(s): *- The Charging Station will empty its Transaction message queue, this will only contain TransactionEventRequests if TxStartPoint was ParkingBayOccupancy and a Transaction was started.*	**6.** The OCTT responds with a **TransactionEventResponse**
Tool validations	* Step 5: Message **TransactionEventRequest** - **offline** must be *true* - **TriggerReason** must be *EVDetected*	
	Post scenario validations: N/a	

Table 108. Test Case Id: TC_C_24_CS

Test case name	**Offline authorization through local authorization list - Expired**	
Test case Id	TC_C_24_CS	
Use case Id(s)	C13	
Requirement(s)	C13.FR.02	
System under test	Charging Station	
Description	This test case describes how to authorize an IdToken via the Local Authorization List while the Charging Station is online. When offline the Charging Station can then locally authorize the IdToken.	
Purpose	To verify if the Charging Station is able to respond correctly to an idToken which has status "Expired" in its local authorization list according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	- LocalAuthListCtrlr.LocalAuthListAvailable is implemented with value true - The Charging Station supports authorization methods other than NoAuthorization	
Before (Preparations)	**Configuration State:** **LocalAuthListEnabled** is *true* (If implemented) **LocalPreAuthorize** is *true* (If implemented) **OfflineTxForUnknownIdEnabled** is *false* (If implemented) **LocalAuthorizeOffline** is *true* **OfflineThreshold** is *<Configured RetryBackOffWaitMinimum_duration> + 60.0* **RetryBackOffWaitMinimum** is *<Configured RetryBackOffWaitMinimum_duration>* **RetryBackOffRandomRange** is *0*	
	Memory State: *IdTokenLocalAuthList* for *<Configured expired idtoken fields>*	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. *The OCTT closes the WebSocket connection AND does not accept a reconnect.*	
	2. Manual Action: *Drive EV into parking bay.*	
	3. Manual Action: *Present idToken.*	
	4. *The OCTT accepts reconnection attempt from the Charging Station.*	
	5. The Charging Stations sends a **TransactionEventRequest** Note(s): - *The Charging Station will empty its Transaction message queue, this will only contain TransactionEventRequests if TxStartPoint was ParkingBayOccupancy and a Transaction was started.*	**6.** The OCTT responds with a **TransactionEventResponse**
Tool validations	* Step 5: Message **TransactionEventRequest** - **offline** must be *true* - **TriggerReason** must be *EVDetected*	
	Post scenario validations: N/a	

Table 109. Test Case Id: TC_C_25_CS

Test case name	**Offline authorization through local authorization list - Local Authorization List > Authorization Cache**	
Test case Id	TC_C_25_CS	
Use case Id(s)	C13, C14	
Requirement(s)	C13.FR.01, C14.FR.01	
System under test	Charging Station	
Description	This test case describes how to authorize an IdToken via the Local Authorization List while the Charging Station is online. When offline the Charging Station can then locally authorize the IdToken.	
Purpose	To verify if the Charging Station does not start a transaction while being offline for an idToken that is stored in the cache, but also in the local authorization list as with status invalid.	
Prerequisite(s)	- LocalAuthListCtrlr.LocalAuthListAvailable is implemented with value true - AuthCacheCtrlr.AuthCacheAvailable is implemented with value true - OfflineTxForUnknownIdEnabled is implemented. - The Charging Station supports authorization methods other than NoAuthorization	
Before (Preparations)	**Configuration State:** **LocalAuthListEnabled** is *true* (If implemented) **LocalPreAuthorize** is *true* (If implemented) **OfflineTxForUnknownIdEnabled** is *true* **LocalAuthorizeOffline** is *true* **StopTxOnInvalidId** is *false* **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>*	
	Memory State: *IdTokenCached <Configured valid idtoken fields>* *IdTokenLocalAuthList* for *<Configured valid idtoken fields, but set as invalid>*	
	Reusable State(s): **State is** *EVConnectedPreSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. *The OCTT closes the WebSocket connection AND does not accept a reconnect.*	
	Manual Action: *Present idToken.*	
	Note(s): *The tool will wait for <Configured Transaction Duration> seconds.*	
	2. *The OCTT accepts reconnection attempt from the Charging Station.*	
	3. The Charging Station does NOT start a transaction. MeterValues are allowed.	
Tool validations	N/a	
	Post scenario validations: N/a	

Table 110. Test Case Id: TC_C_26_CS

Test case name	**Offline Authorization - Unknown Id**	
Test case Id	TC_C_26_CS	
Use case Id(s)	C15 & C13	
Requirement(s)	C15.FR.02,C15.FR.06,C15.FR.08,C13.FR.04	
System under test	Charging Station	
Description	The Charging Station is allowed to allow starting a transaction for unknown idTokens when offline and configured to do so.	
Purpose	To verify if the Charging Station is able to start a transaction while being offline for an unknown idToken, when it is configured to do so.	
Prerequisite(s)	- OfflineTxForUnknownIdEnabled is implemented. - The Charging Station supports authorization methods other than NoAuthorization	
Before (Preparations)	**Configuration State:** **LocalAuthListEnabled** is *true* (If implemented) **LocalPreAuthorize** is *true* (If implemented) **OfflineTxForUnknownIdEnabled** is *true* **LocalAuthorizeOffline** is *true* **MaxEnergyOnInvalidId** is *0* (If implemented) **StopTxOnInvalidId** is *false*	
	Memory State: N/a	
	Reusable State(s): **State is** *StartOfflineTransaction*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The Charging Stations sends a **TransactionEventRequest** Note(s): *- The Charging Station will empty its Transaction message queue. This will contain one or more TransactionEventRequest messages*	**2.** The OCTT responds with a **TransactionEventResponse** Note(s): *- The OCTT will respond to the TransactionEventRequest containing the idToken, with* ***idtokenInfo.status*** *Invalid*
	Manual Action: *Present valid idToken.*	
	Manual Action: *Unplug cable*	
	3. The Charging Stations sends a **TransactionEventRequest** with **triggerReason** *StopAuthorized*	**4.** The OCTT responds with a **TransactionEventResponse**
Tool validations	* Step 1: All Message(s): **TransactionEventRequest** - **offline** must be *true* * Step 1: One of the Message(s): **TransactionEventRequest** - **chargingState** must be *SuspendedEVSE*	
	Post scenario validations: N/a	

Table 111. Test Case Id: TC_C_50_CS

Test case name	**Authorization using Contract Certificates 15118 - Online - Local contract certificate validation - Accepted**	
Test case Id	TC_C_50_CS	
Use case Id(s)	C07	
Requirement(s)	C07.FR.01,C07.FR.02	
System under test	Charging Station	
Description	The Charging Station is able to authorize with contract certificates when it supports ISO 15118.	
Purpose	To verify if the Charging Station is able to authorize while locally validating the contract certificate.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** **AuthCtrlr.AuthEnabled** is *true* (If implemented AND ReadWrite) **AuthCtrlr.DisableRemoteAuthorization** is *false* (If implemented) For the ISO15118Ctrlr of the EVSE used in the PnC transaction: **ISO15118Ctrlr.CentralContractValidationAllowed** is *false* **ISO15118Ctrlr.ContractCertificateInstallationEnabled** is *true* **ISO15118Ctrlr.V2GCertificateInstallationEnabled** is *true* **ISO15118Ctrlr.PnCEnabled** is *true* **ISO15118Ctrlr.SeccId** is configured seccId **ISO15118Ctrlr.CountryName** is *NL* **ISO15118Ctrlr.OrganizationName** is configured vendorId	
	Memory State: *CertificateInstalled* for certificateType *V2GRootCertificate* *CertificateInstalled* for certificateType *MORootCertificate*	
	Reusable State(s): **State is** *EVConnectedPreSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The Charging Station sends an **AuthorizeRequest** Note(s): *-The test case should be robust enough to also handle a* **GetCertificateStatusRequest**	**2.** The OCTT responds with a **AuthorizeResponse** with **idTokenInfo.status** *Accepted* and **certificateStatus** = *Accepted*
	3. The Charging Station sends a **TransactionEventRequest** Note(s): *- This step needs to be executed when* ***TxStartPoint*** *contains Authorized OR the transaction already started. So in the case* ***TxStartPoint*** *contains ParkingBayOccupancy or (EVConnected, in the case this testcase was initiated from state EVConnectedPreSession.)*	**4.** The OCTT responds with a **TransactionEventResponse** With **idTokenInfo.status** *Accepted*
	5. Execute **Reusable State** *EnergyTransferStarted*	
Tool validations	* Step 1: Message: **AuthorizeRequest** - **idToken.type** must be *eMAID* - **iso15118CertificateHashData** is provided * Step 3: Message: **TransactionEventRequest** - **triggerReason** must be *Authorized*	
	Post scenario validations: N/a	

Table 112. Test Case Id: TC_C_51_CS

Test case name	**Authorization using Contract Certificates 15118 - Online - Local contract certificate validation - Rejected**	
Test case Id	TC_C_51_CS	
Use case Id(s)	C07	
Requirement(s)	C07.FR.01,C07.FR.02	
System under test	Charging Station	
Description	The Charging Station is able to authorize with contract certificates when it supports ISO 15118.	
Purpose	To verify if the Charging Station is able to handle a rejected on an AuthorizeRequest, when authorizing using a contract certificate with an invalid EMAID.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** **AuthCtrlr.AuthEnabled** is *true* (If implemented AND ReadWrite) **AuthCtrlr.DisableRemoteAuthorization** is *false* (If implemented) For the ISO15118Ctrlr of the EVSE used in the PnC transaction: **ISO15118Ctrlr.CentralContractValidationAllowed** is *false* **ISO15118Ctrlr.PnCEnabled** is *true*	
	Memory State: *CertificateInstalled* for certificateType *V2GRootCertificate* *CertificateInstalled* for certificateType *MORootCertificate*	
	Reusable State(s): **State is** *EVConnectedPreSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The Charging Station sends an **AuthorizeRequest** Note(s): *-The test case should be robust enough to also handle a* **GetCertificateStatusRequest**	**2.** The OCTT responds with a **AuthorizeResponse** with **idTokenInfo.status** *Invalid* and **certificateStatus** = *ContractCancelled*
Tool validations	* Step 1: Message: **AuthorizeRequest** - **idToken.type** must be *eMAID* - **iso15118CertificateHashData** is provided	
	Post scenario validations: EV is not authorized and shall not charge: Charging Station does not send **TransactionEventRequest** with: - **triggerReason** = *Authorized* or **chargingState** = *Charging*	

Table 113. Test Case Id: TC_C_52_CS

Test case name	**Authorization using Contract Certificates 15118 - Online - Central contract certificate validation - Accepted**	
Test case Id	TC_C_52_CS	
Use case Id(s)	C07	
Requirement(s)	C07.FR.01,C07.FR.02,C07.FR.06	
System under test	Charging Station	
Description	The Charging Station is able to authorize with contract certificates when it supports ISO 15118.	
Purpose	To verify if the Charging Station is able to authorize, while not being able to locally validate the contract certificate and then send it to the CSMS.	
Prerequisite(s)	- The V2G/MO Root certificate that is needed to validate the EV Contract certificate must NOT be installed at the Charging Station. - The Charging Station supports central contract validation.	
Before (Preparations)	**Configuration State:** **AuthCtrlr.AuthEnabled** is *true* (If implemented AND ReadWrite) **AuthCtrlr.DisableRemoteAuthorization** is *false* (If implemented) For the ISO15118Ctrlr of the EVSE used for the PnC transaction: **ISO15118Ctrlr.CentralContractValidationAllowed** is *true* **ISO15118Ctrlr.ContractCertificateInstallationEnabled** is *true* **ISO15118Ctrlr.V2GCertificateInstallationEnabled** is *true* **ISO15118Ctrlr.PnCEnabled** is *true* **ISO15118Ctrlr.SeccId** is configured seccId **ISO15118Ctrlr.CountryName** is *NL* **ISO15118Ctrlr.OrganizationName** is configured vendorId	
	Memory State: N/a	
	Reusable State(s): **State is** *EVConnectedPreSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The Charging Station sends an **AuthorizeRequest** Note(s): *-The test case should be robust enough to also handle a* **GetCertificateStatusRequest**	**2.** The OCTT responds with a **AuthorizeResponse** with **idTokenInfo.status** *Accepted* and **certificateStatus** = *Accepted*
	3. The Charging Station sends a **TransactionEventRequest** Note(s): *- This step needs to be executed when* ***TxStartPoint*** *contains Authorized OR the transaction already started. So in the case* ***TxStartPoint*** *contains ParkingBayOccupancy or (EVConnected, in the case this testcase was initiated from state EVConnectedPreSession.)*	**4.** The OCTT responds with a **TransactionEventResponse** With **idTokenInfo.status** *Accepted*
	5. Execute **Reusable State** *EnergyTransferStarted*	
Tool validations	* Step 1: Message: **AuthorizeRequest** - **idToken.type** must be *eMAID* - **iso15118CertificateHashData** may be provided - **certificate** is provided * Step 3: Message: **TransactionEventRequest** - **triggerReason** must be *Authorized*	
	Post scenario validations: N/a	

Table 114. Test Case Id: TC_C_53_CS

Test case name	**Authorization using Contract Certificates 15118 - Online - Central contract validation fails**	
Test case Id	TC_C_53_CS	
Use case Id(s)	C07	
Requirement(s)	N/a	
System under test	Charging Station	
Description	The Charging Station is able to authorize with contract certificates when it supports ISO 15118.	
Purpose	To verify if the Charging Station is able to handle an invalid contract certificate.	
Prerequisite(s)	- The V2G/MO Root certificate that is needed to validate the EV Contract certificate must NOT be installed at the Charging Station. - The Charging Station supports central contract validation.	
Before (Preparations)	**Configuration State:** **AuthCtrlr.AuthEnabled** is *true* (If implemented AND ReadWrite) **AuthCtrlr.DisableRemoteAuthorization** is *false* (If implemented) For the ISO15118Ctrlr of the EVSE involved in the PnC transaction: **ISO15118Ctrlr.CentralContractValidationAllowed** is *true* **ISO15118Ctrlr.PnCEnabled** is *true*	
	Memory State: N/a	
	Reusable State(s): **State is** *EVConnectedPreSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The Charging Station sends an **AuthorizeRequest**. Note(s): *-The test case should be robust enough to also handle a* **GetCertificateStatusRequest**	**2.** The OCTT responds with a **AuthorizeResponse** with **idTokenInfo.status** *Invalid* and **certificateStatus** = *CertificateRevoked*
Tool validations	* Step 1: Message: **AuthorizeRequest** - **idToken.type** must be *eMAID* - **iso15118CertificateHashData** may be provided - **certificate** is provided	
	Post scenario validations: EV is not authorized and shall not charge: Charging Station does not send **TransactionEventRequest** with: - **triggerReason** = *Authorized* or **chargingState** = *Charging*	

Table 115. Test Case Id: TC_C_54_CS

Test case name	**Authorization using Contract Certificates 15118 - Offline - ContractValidationOffline is true**	
Test case Id	TC_C_54_CS	
Use case Id(s)	C07	
Requirement(s)	C07.FR.08,C07.FR.09,C07.FR.10,C07.FR.11,C07.FR.12	
System under test	Charging Station	
Description	The Charging Station is able to authorize with contract certificates when it supports ISO 15118.	
Purpose	To verify if the Charging Station is able to authorize using contract certificates, while it is offline.	
Prerequisite(s)	The Charging Station supports authorization methods other than NoAuthorization	
Before (Preparations)	**Configuration State:** **AuthCtrlr.AuthEnabled** is *true* (If implemented AND ReadWrite) **OfflineTxForUnknownIdEnabled** is *true* (If implemented) **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>* For ISO15118Ctrlr of EVSE involved in PnC transaction: **ISO15118Ctrlr.ContractValidationOffline** is *true* **ISO15118Ctrlr.PnCEnabled** is *true*	
	Memory State: *CertificateInstalled* for certificateType *V2GRootCertificate* *CertificateInstalled* for certificateType *MORootCertificate* *IdTokenCached* for *<Configured valid IdToken fields>* (If implemented) *IdTokenLocalAuthList* for *<Configured valid IdToken fields>* (If implemented)	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. *The OCTT closes the WebSocket connection AND does not accept a reconnect.*	
	Manual Action: *Drive EV into parking bay.*	
	Manual Action: *Connect the EV and EVSE.*	
	Notes(s): *The tool will wait for <Configured Transaction Duration> seconds.*	
	2. *The OCTT accepts the reconnection attempt from the Charging Station after <Configured Transaction Duration> seconds.*	
	3. The Charging Station notifies the CSMS about the status change of the connector.	**4.** The OCTT responds accordingly.
	5. The Charging Station sends a **TransactionEventRequest** Note(s): *- This step needs to be executed when* ***TxStartPoint*** *contains EVConnected OR the transaction already started. So in the case* ***TxStartPoint*** *contains ParkingBayOccupancy*	**6.** The OCTT responds with a **TransactionEventResponse**
	7. The Charging Station sends a **TransactionEventRequest** Note(s): *- This step needs to be executed when* ***TxStartPoint*** *contains Authorized OR the transaction already started. So in the case* ***TxStartPoint*** *contains ParkingBayOccupancy or EVConnected.*	**8.** The OCTT responds with a **TransactionEventResponse** With **idTokenInfo.status** *Accepted*
	9. Execute **Reusable State** *EnergyTransferStarted*	

Test case name	**Authorization using Contract Certificates 15118 - Offline - ContractValidationOffline is true**
Tool validations	* Step 3: Message: **StatusNotificationRequest** - **connectorStatus** must be *Occupied* Message: **NotifyEventRequest** - **eventData[0].trigger** must be *Delta* - **eventData[0].actualValue** must be *Occupied* - **eventData[0].component.name** must be *Connector* - **eventData[0].variable.name** must be *AvailabilityState* * Step 5: Message: **TransactionEventRequest** - **triggerReason** must be *CablePluggedIn* - **transactionInfo.chargingState** must be *EVConnected* - **offline** *true* * Step 7: Message: **TransactionEventRequest** - **triggerReason** must be *Authorized* - **offline** *true*
	Post scenario validations: N/a

Table 116. Test Case Id: TC_C_55_CS

Test case name	**Authorization using Contract Certificates 15118 - Offline - ContractValidationOffline is false**	
Test case Id	TC_C_55_CS	
Use case Id(s)	C07	
Requirement(s)	C07.FR.07	
System under test	Charging Station	
Description	The Charging Station is able to authorize with contract certificates when it supports ISO 15118.	
Purpose	To verify if the Charging Station is able to handle being offline and not allowing a charging session to start, when it is configured to do so.	
Prerequisite(s)	The Charging Station supports authorization methods other than NoAuthorization	
Before (Preparations)	**Configuration State:** **AuthCtrlr.AuthEnabled** is *true* (If implemented AND ReadWrite) **OfflineTxForUnknownIdEnabled** is *true* (If implemented) **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>* For the ISO15118Ctrlr of the EVSE involved in the PnC transaction: **ISO15118Ctrlr.ContractValidationOffline** is *false* **ISO15118Ctrlr.PnCEnabled** is *true*	
	Memory State: *CertificateInstalled* for certificateType *V2GRootCertificate* *CertificateInstalled* for certificateType *MORootCertificate* *IdTokenCached* for *<Configured valid IdToken fields>* (If implemented) *IdTokenLocalAuthList* for *<Configured valid IdToken fields>* (If implemented)	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. *The OCTT closes the WebSocket connection AND does not accept a reconnect.*	
	Manual Action: *Drive EV into parking bay.*	
	Manual Action: *Connect the EV and EVSE.*	
	Note(s): *The tool will wait for <Configured Transaction Duration> seconds.*	
	2. *The OCTT accepts the reconnection attempt from the Charging Station after <Configured Transaction Duration> seconds.*	
	3. The Charging Station notifies the CSMS about the status change of the connector.	**4.** The OCTT responds accordingly.
	5. The Charging Station sends a **TransactionEventRequest** Note(s): *- This step needs to be executed when **TxStartPoint** contains EVConnected OR the transaction already started. So in the case **TxStartPoint** contains ParkingBayOccupancy*	**6.** The OCTT responds with a **TransactionEventResponse**
	7. The Charging Station has NOT started charging and does NOT send **TransactionEventRequest** message(s) with triggerReason *Authorized* OR *ChargingStateChanged*.	

Test case name	**Authorization using Contract Certificates 15118 - Offline - ContractValidationOffline is false**
Tool validations	* Step 3: Message: **StatusNotificationRequest** - **connectorStatus** must be *Occupied* Message: **NotifyEventRequest** - **eventData[0].trigger** must be *Delta* - **eventData[0].actualValue** must be *Occupied* - **eventData[0].component.name** must be *Connector* - **eventData[0].variable.name** must be *AvailabilityState* * Step 5: Message: **TransactionEventRequest** - **triggerReason** must be *CablePluggedIn* - **transactionInfo.chargingState** must be *EVConnected* - **offline** *true*
	Post scenario validations: EV is not authorized and shall not charge: Charging Station does not send **TransactionEventRequest** with: - **triggerReason** = *Authorized* or **chargingState** = *Charging*

Table 117. Test Case Id: TC_C_56_CS

Test case name	**Local start transaction - Authorization Unknown**	
Test case Id	TC_C_56_CS	
Use case Id(s)	C01	
Requirement(s)	C01.FR.02	
System under test	Charging Station	
Description	When a Charging Station needs to charge an EV, it needs to authorize the EV Driver first at the CSMS before the charging can be started or stopped.	
Purpose	To verify whether the Charging Station is able to handle receiving an Unknown idToken.	
Prerequisite(s)	The Charging Station does NOT have a cable lock, which prevents the EV driver to connect the EV and EVSE before authorization. The Charging Station supports authorization methods other than NoAuthorization	
Before (Preparations)	**Configuration State:** **AuthCtrlr.AuthEnabled** is *true* (If implemented AND ReadWrite) **AuthCtrlr.DisableRemoteAuthorization** is *false* (If implemented)	
	Memory State: N/a	
	Reusable State(s): **State is** *EVConnectedPreSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Present invalid idToken.*	
	1. The Charging Station sends an **AuthorizeRequest**	**2.** The OCTT responds with an **AuthorizeResponse** with **idTokenInfo.status** *Unknown*
	Note(s): - *The Charging Station SHALL NOT send a TransactionEventRequest message after the AuthorizeRequest from step 1.* - *The OCTT waits <Configured message timeout> seconds, before ending the testcase.*	
Tool validations	* Step 1: Message: **AuthorizeRequest** - **idToken.idToken** *<Configured invalid_idtoken_idtoken>* - **idToken.type** *<Configured invalid_idtoken_type>*	
	Post scenario validations: N/a	

2.5. D Local Authorization List Management

Table 118. Test Case Id: TC_D_01_CS

Test case name	**Send Local Authorization List - Full**	
Test case Id	TC_D_01_CS	
Use case Id(s)	D01	
Requirement(s)	D01_FR_02, D01_FR_15	
System under test	Charging Station	
Description	The CSMS sends a Local Authorization List which a Charging Station can use for the authorization of idTokens. The list MAY be either a full list to replace the current list in the Charging Station or it MAY be a differential list with updates to be applied to the current list in the Charging Station.	
Purpose	To verify if the Charging Station is able to replace the Local Authorization List according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	The Charging Station supports the Local Authorization List feature	
Before (Preparations)	**Configuration State:** **LocalAuthListEnabled** is *true* (If implemented) *Configured versionNumber > 0	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SendLocalListResponse**	**1.** The OCTT sends a **SendLocalListRequest** with - **updateType** *Full* - **versionNumber** *<Configured versionNumber>* - **localAuthorizationList[0].idToken.idToken** *<Configured valid_idtoken_idtoken>* - **localAuthorizationList[0].idToken.type** *<Configured valid_idtoken_type>*
	4. The Charging Station responds with a **GetLocalListVersionResponse**	**3.** The OCTT sends a **GetLocalListVersionRequest**
Tool validations	* Step 2: Message **SendLocalListResponse** - **status** *Accepted* * Step 4: Message **GetLocalListVersionResponse** - **versionNumber** *<Equal to version sent in step 1>*	
	Post scenario validations: - N/a	

Table 119. Test Case Id: TC_D_02_CS

Test case name	**Send Local Authorization List - Differential Update**	
Test case Id	TC_D_02_CS	
Use case Id(s)	D01	
Requirement(s)	D01_FR_02, D01_FR_16	
System under test	Charging Station	
Description	The CSMS sends a Local Authorization List which a Charging Station can use for the authorization of idTokens. The list MAY be either a full list to replace the current list in the Charging Station or it MAY be a differential list with updates to be applied to the current list in the Charging Station.	
Purpose	To verify if the Charging Station is able to replace the Local Authorization List in differential type according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	The Charging Station supports the Local Authorization List feature	
Before (Preparations)	**Configuration State:** **LocalAuthListEnabled** is *true* (If implemented) *Configured versionNumber > 0	
	Memory State: *IdTokenLocalAuthList* for *<Configured valid idtoken fields>*	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SendLocalListResponse**	**1.** The OCTT sends a **SendLocalListRequest** with - **updateType** *Differential* - **versionNumber** *<Configured versionNumber + 1>* - **localAuthorizationList[0].idToken.idToken** *<Configured valid_idtoken_idtoken2>* - **localAuthorizationList[0].idToken.type** *<Configured valid_idtoken_type2>*
	Note(s): *The message send by OCTT is within ItemsPerMessageSendLocalList AND BytesPerMessageSendLocalList range.*	
	4. The Charging Station responds with a **GetLocalListVersionResponse**	**3.** The OCTT sends a **GetLocalListVersionRequest**
Tool validations	* Step 2: Message **SendLocalListResponse** - **status** *Accepted* * Step 4: Message **GetLocalListVersionResponse** - **versionNumber** *<Equal to version send in step 1>*	
	Post scenario validations: - N/a	

Table 120. Test Case Id: TC_D_03_CS

Test case name	**Send Local Authorization List - Differential Remove**	
Test case Id	TC_D_03_CS	
Use case Id(s)	D01	
Requirement(s)	D01_FR_02, D01_FR_17	
System under test	Charging Station	
Description	The CSMS sends a Local Authorization List which a Charging Station can use for the authorization of idTokens. The list MAY be either a full list to replace the current list in the Charging Station or it MAY be a differential list with updates to be applied to the current list in the Charging Station.	
Purpose	To verify if the Charging Station is able to remove items from the Local Authorization List when send in differential type with data without idToken according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	The Charging Station supports the Local Authorization List feature	
Before (Preparations)	**Configuration State:** **LocalAuthListEnabled** is *true* (If implemented) *Configured versionNumber > 0	
	Memory State: *IdTokenLocalAuthList* for <*Configured valid idtoken fields*>	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SendLocalListResponse**	**1.** The OCTT sends a **SendLocalListRequest** with - **updateType** *Differential* - **versionNumber** <*Configured versionNumber + 1*> - **localAuthorizationList** <*Contains AuthorizationData elements without idTokenInfo*>
	Note(s): *The message send by OCTT is within ItemsPerMessageSendLocalList AND BytesPerMessageSendLocalList range.*	
	4. The Charging Station responds with a **GetLocalListVersionResponse**	**3.** The OCTT sends a **GetLocalListVersionRequest**
Tool validations	* Step 2: Message **SendLocalListResponse** - **status** *Accepted* * Step 3: Message **GetLocalListVersionResponse** - **versionNumber** <*Equal to version sent in step 1*>	
	Post scenario validations: - N/a	

Table 121. Test Case Id: TC_D_04_CS

Test case name	**Send Local Authorization List - Full with empy list**	
Test case Id	TC_D_04_CS	
Use case Id(s)	D01	
Requirement(s)	D01_FR_02, D01_FR_04	
System under test	Charging Station	
Description	The CSMS sends a Local Authorization List which a Charging Station can use for the authorization of idTokens. The list MAY be either a full list to replace the current list in the Charging Station or it MAY be a differential list with updates to be applied to the current list in the Charging Station.	
Purpose	To verify if the Charging Station is able to remove all items from the Local Authorization List when send in full type with no data according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	The Charging Station supports the Local Authorization List feature	
Before (Preparations)	**Configuration State:** **LocalAuthListEnabled** is *true* (If implemented) *Configured versionNumber > 0	
	Memory State: *IdTokenLocalAuthList* for *<Configured valid idtoken fields>*	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SendLocalListResponse**	**1.** The OCTT sends a **SendLocalListRequest** with - **updateType** *Full* - **versionNumber** *<Configured versionNumber>* - **localAuthorizationList** *<Empty>*
	4. The Charging Station responds with a **GetLocalListVersionResponse**	**3.** The OCTT sends a **GetLocalListVersionRequest**
Tool validations	* Step 2: Message **SendLocalListResponse** - **status** *Accepted* * Step 4: Message **GetLocalListVersionResponse** - **versionNumber** *<Configured versionNumber>*	
	Post scenario validations: - N/a	

Table 122. Test Case Id: TC_D_05_CS

Test case name	**Send Local Authorization List - Differential with empty list**	
Test case Id	TC_D_05_CS	
Use case Id(s)	D01	
Requirement(s)	D01_FR_02, D01_FR_05	
System under test	Charging Station	
Description	The CSMS sends a Local Authorization List which a Charging Station can use for the authorization of idTokens. The list MAY be either a full list to replace the current list in the Charging Station or it MAY be a differential list with updates to be applied to the current list in the Charging Station.	
Purpose	To verify if the Charging Station is able to correctly respond on a Local Authorization List when send in differential type with no data according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	The Charging Station supports the Local Authorization List feature	
Before (Preparations)	**Configuration State:** **LocalAuthListEnabled** is *true* (If implemented) *Configured versionNumber > 0	
	Memory State: *IdTokenLocalAuthList* for *<Configured valid idtoken fields>*	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SendLocalListResponse**	**1.** The OCTT sends a **SendLocalListRequest** with - **updateType** *Differential* - **versionNumber** *<Configured versionNumber + 1>* - **localAuthorizationList** *<Empty>*
	4. The Charging Station responds with a **GetLocalListVersionResponse**	**3.** The OCTT sends a **GetLocalListVersionRequest**
Tool validations	* Step 2: Message **SendLocalListResponse** - **status** *Accepted* * Step 4: Message **GetLocalListVersionResponse** - **versionNumber** *<Equal to the version send in step 1>*	
	Post scenario validations: - N/a	

Table 123. Test Case Id: TC_D_06_CS

Test case name	**Send Local Authorization List - VersionMismatch**	
Test case Id	TC_D_06_CS	
Use case Id(s)	D01	
Requirement(s)	D01_FR_19	
System under test	Charging Station	
Description	The CSMS sends a Local Authorization List which a Charging Station can use for the authorization of idTokens. The list MAY be either a full list to replace the current list in the Charging Station or it MAY be a differential list with updates to be applied to the current list in the Charging Station.	
Purpose	To verify if the Charging Station is able to correctly respond on a Local Authorization List when send in differential type with with a faulty version number according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	The Charging Station supports the Local Authorization List feature	
Before (Preparations)	**Configuration State:** **LocalAuthListEnabled** is *true* (If implemented) *Configured versionNumber > 1	
	Memory State: *IdTokenLocalAuthList* for *<Configured valid idtoken fields>*	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SendLocalListResponse**	**1.** The OCTT sends a **SendLocalListRequest** with - **updateType** *Differential* - **versionNumber** *<Configured versionNumber - 1>* - **localAuthorizationList** *<Not Empty>*
	4. The Charging Station responds with a **GetLocalListVersionResponse**	**3.** The OCTT sends a **GetLocalListVersionRequest**
Tool validations	* Step 2: Message **SendLocalListResponse** - **status** *VersionMismatch* * Step 4: Message **GetLocalListVersionResponse** - **versionNumber** *<Configured versionNumber>*	
	Post scenario validations: - N/a	

Table 124. Test Case Id: TC_D_07_CS

Test case name	**Send Local Authorization List - Persistent over reboot**	
Test case Id	TC_D_07_CS	
Use case Id(s)	D01	
Requirement(s)	D01_FR_10	
System under test	Charging Station	
Description	The CSMS sends a Local Authorization List which a Charging Station can use for the authorization of idTokens. The list MAY be either a full list to replace the current list in the Charging Station or it MAY be a differential list with updates to be applied to the current list in the Charging Station.	
Purpose	To verify if the Charging Station is able to save the Local Authorization List persistent over reboot according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	The Charging Station supports the Local Authorization List feature and stores it in non-volatile memory	
Before (Preparations)	**Configuration State:** **LocalAuthListEnabled** is *true* (If implemented) *Configured versionNumber > 0	
	Memory State: *IdTokenLocalAuthList* for *<Configured valid idtoken fields>*	
	Charging State: *Booted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **GetLocalListVersionResponse**	**1.** The OCTT sends a **GetLocalListVersionRequest**
Tool validations	* Step 2: Message **GetLocalListVersionResponse** - **versionNumber** *<Configured versionNumber>*	
	Post scenario validations: - N/a	

Table 125. Test Case Id: TC_D_08_CS

Test case name	**Get Local List Version - Success**	
Test case Id	TC_D_08_CS	
Use case Id(s)	D02	
Requirement(s)	D02_FR_01	
System under test	Charging Station	
Description	The CSMS can request a Charging Station for the version number of the Local Authorization List by sending a GetLocalListVersionRequest.	
Purpose	To verify if the Charging Station is able to respond the Local Authorization List version number according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	The Charging Station supports the Local Authorization List feature	
Before (Preparations)	**Configuration State:** **LocalAuthListEnabled** is *true* (If implemented) *Configured versionNumber > 0	
	Memory State: *IdTokenLocalAuthList* for *<Configured valid idtoken fields>*	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
		1. The OCTT sends a **GetLocalListVersionRequest**
	2. The Charging Station responds with a **GetLocalListVersionResponse**	
Tool validations	* Step 2: Message **GetLocalListVersionResponse** - **versionNumber** *<Configured versionNumber>*	
	Post scenario validations: - N/a	

Table 126. Test Case Id: TC_D_10_CS

Test case name	**Get Local List Version - Function disabled**	
Test case Id	TC_D_10_CS	
Use case Id(s)	D02	
Requirement(s)	D02_FR_03	
System under test	Charging Station	
Description	The CSMS can request a Charging Station for the version number of the Local Authorization List by sending a GetLocalListVersionRequest.	
Purpose	To verify if the Charging Station is able to respond the Local Authorization List version number according to the mechanism as described in the OCPP specification when the **LocalAuthListEnabled** is set to false.	
Prerequisite(s)	The Charging Station supports the Local Authorization List feature	
Before (Preparations)	**Configuration State:** **LocalAuthListEnabled** is *false* (If implemented)	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **GetLocalListVersionResponse**	**1.** The OCTT sends a **GetLocalListVersionRequest**
Tool validations	* Step 2: Message **GetLocalListVersionResponse** - **versionNumber** *0*	
	Post scenario validations: - N/a	

2.6. E Transactions

Table 127. Test Case Id: TC_E_01_CS

Test case name	**Start transaction options - PowerPathClosed**	
Test case Id	TC_E_01_CS	
Use case Id(s)	E01(S5)	
Requirement(s)	E01.FR.05, E01.FR.07, E01.FR.10, E01.FR.15, E01.FR.16	
System under test	Charging Station	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the start options.	
Purpose	To verify if the Charging Station starts a transaction when the power path has been closed and it has been configured to do so.	
Prerequisite(s)	- The Charging Station does NOT have the following configuration; The mutability of **TxStartPoint** is *ReadOnly* AND (the value *PowerPathClosed* is NOT set OR (*ParkingBayOccupancy* OR *EVConnected* OR *Authorized* OR *DataSigned*), is set). - If the mutability of **TxStartPoint** is *ReadWrite*, then the value *PowerPathClosed* must be supported.	
Before (Preparations)	**Configuration State:** If the mutability of **TxStartPoint** is *ReadWrite* then **TxStartPoint** contains *PowerPathClosed*	
	Memory State: N/a	
	Reusable State(s): **State is** *Authorized*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *EVConnectedPreSession*	
Tool validations	N/a	
	Post scenario validations: N/a	

Table 128. Test Case Id: TC_E_02_CS

Test case name	**Start transaction options - EnergyTransfer**	
Test case Id	TC_E_02_CS	
Use case Id(s)	E01(S6)	
Requirement(s)	E01.FR.06, E01.FR.07, E01.FR.10, E01.FR.15, E01.FR.16	
System under test	Charging Station	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the start options.	
Purpose	To verify if the Charging Station starts a transaction when the energy transfer starts and it has been configured to do so.	
Prerequisite(s)	- The Charging Station does NOT have the following configuration; The mutability of **TxStartPoint** is *ReadOnly* AND (the value *EnergyTransfer* is NOT set OR (*ParkingBayOccupancy* OR *EVConnected* OR *Authorized* OR *DataSigned* OR *PowerPathClosed*), is set). - If the mutability of **TxStartPoint** is *ReadWrite*, then the value *EnergyTransfer* must be supported.	
Before (Preparations)	**Configuration State:** If the mutability of **TxStartPoint** is *ReadWrite* then **TxStartPoint** contains *EnergyTransfer*	
	Memory State: N/a	
	Reusable State(s): **State is** *Authorized*	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Connect the EV and EVSE.*	
	1. The Charging Station notifies the CSMS about the status change of the connector.	**2.** The OCTT responds accordingly.
	3. The Charging Station sends a **TransactionEventRequest**	**4.** The OCTT responds with a **TransactionEventResponse**
Tool validations	* Step 1: Message: **StatusNotificationRequest** - **connectorStatus** must be *Occupied* Message: **NotifyEventRequest** - **eventData[0].trigger** must be *Delta* - **eventData[0].actualValue** must be *Occupied* - **eventData[0].component.name** must be *Connector* - **eventData[0].variable.name** must be *AvailabilityState* * Step 3: Message: **TransactionEventRequest** - **eventType** must be *Started* - If the OCTT is configured to start transactions using a RequestStartTransactionRequest message then **triggerReason** must be *RemoteStart* Else **triggerReason** must be *ChargingStateChanged* or *Authorized* - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>* - **evse** must be provided - **evse.connectorId** must be provided - **transactionInfo.chargingState** must be *Charging*	
	Post scenario validations: N/a	

Table 129. Test Case Id: TC_E_09_CS

Test case name	**Start transaction options - EVConnected**	
Test case Id	TC_E_09_CS	
Use case Id(s)	E01(S2)	
Requirement(s)	E01.FR.02, E01.FR.07, E01.FR.10, E01.FR.15, E01.FR.16	
System under test	Charging Station	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the start options.	
Purpose	To verify if the Charging Station starts a transaction when the EV and EVSE are connected and it has been configured to do so.	
Prerequisite(s)	- The Charging Station does NOT have the following configuration; The mutability of **TxStartPoint** is *ReadOnly* AND (the value *EVConnected* is NOT set OR *ParkingBayOccupancy* is set). - If the mutability of **TxStartPoint** is *ReadWrite*, then the value *EVConnected* must be supported.	
Before (Preparations)	**Configuration State:** If the mutability of **TxStartPoint** is *ReadWrite* then **TxStartPoint** contains *EVConnected*	
	Memory State: N/a	
	Reusable State(s): **State is** *ParkingBayOccupied*	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Connect the EV and EVSE.*	
	1. The Charging Station notifies the CSMS about the status change of the connector.	**2.** The OCTT responds accordingly.
	3. The Charging Station sends a **TransactionEventRequest**	**4.** The OCTT responds with a **TransactionEventResponse**
Tool validations	* Step 1: Message: **StatusNotificationRequest** - **connectorStatus** must be *Occupied* Message: **NotifyEventRequest** - **eventData[0].trigger** must be *Delta* - **eventData[0].actualValue** must be *Occupied* - **eventData[0].component.name** must be *Connector* - **eventData[0].variable.name** must be *AvailabilityState* * Step 3: Message: **TransactionEventRequest** - **eventType** must be *Started* - **triggerReason** must be *CablePluggedIn* or *ChargingStateChanged* - **evse** must be provided - **evse.connectorId** must be provided - **transactionInfo.chargingState** must be *EVConnected*	
	Post scenario validations: N/a	

Table 130. Test Case Id: TC_E_10_CS

Test case name	**Start transaction options - Authorized - Local**	
Test case Id	TC_E_10_CS	
Use case Id(s)	E01(S3) AND (C01 OR C02 OR C04 OR C06)	
Requirement(s)	E01.FR.03, E01.FR.07, E01.FR.10, E01.FR.15, E01.FR.16 AND C01.FR.02, C02.FR.01, C06.FR.02	
System under test	Charging Station	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the start options.	
Purpose	To verify if the Charging Station starts a transaction when the EV and EVSE are connected and it has been configured to do so.	
Prerequisite(s)	- The Charging Station supports at least one authorization method described at the following Use cases; C01, C02, C04, C06. - The Charging Station does NOT have the following configuration; The mutability of **TxStartPoint** is *ReadOnly* AND the value *Authorized* is NOT set. - If the mutability of **TxStartPoint** is *ReadWrite*, then the value *Authorized* must be supported.	
Before (Preparations)	**Configuration State:** If the mutability of **TxStartPoint** is *ReadWrite* then **TxStartPoint** contains *Authorized* **AuthCtrlr.AuthEnabled** is *true* (If implemented AND ReadWrite) **AuthCtrlr.DisableRemoteAuthorization** is *false* (If implemented)	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	<u>Manual Action</u>: *Present IdToken.*	
	1. The Charging Station sends an **AuthorizeRequest** <u>Note(s)</u>: *- This step needs to be executed, unless **AuthEnabled** is implemented with mutability ReadOnly AND the value is set to false OR a start button as described at Use case C02 is used (This must be configured at the OCTT).*	**2.** The OCTT responds with a **AuthorizeResponse** with **idTokenInfo.status** *Accepted*
	3. The Charging Station sends a **TransactionEventRequest**	**4.** The OCTT responds with a **TransactionEventResponse**
Tool validations	* Step 1: Message: **AuthorizeRequest** - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>* * Step 3: Message: **TransactionEventRequest** - **eventType** must be *Started* - **triggerReason** must be *Authorized* - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>*	
	Post scenario validations: N/a	

Table 131. Test Case Id: TC_E_13_CS

Test case name	**Start transaction options - Authorized - Remote**	
Test case Id	TC_E_13_CS	
Use case Id(s)	E01(S3) AND F02	
Requirement(s)	E01.FR.03 AND F01.FR.03, F01.FR.04, F01.FR.06, F01.FR.19, F02.FR.01	
System under test	Charging Station	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the start options.	
Purpose	To verify if the Charging Station starts a transaction when the EV and EVSE are connected and it has been configured to do so.	
Prerequisite(s)	- The Charging Station does NOT have the following configuration; The mutability of **TxStartPoint** is *ReadOnly* AND the value *Authorized* is NOT set. - If the mutability of **TxStartPoint** is *ReadWrite*, then the value *Authorized* must be supported.	
Before (Preparations)	**Configuration State:** If the mutability of **TxStartPoint** is *ReadWrite* then **TxStartPoint** contains *Authorized* **AuthCtrlr.AuthEnabled** is *true* (If implemented AND ReadWrite) **AuthCtrlr.DisableRemoteAuthorization** is *false* (If implemented)	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **RequestStartTransactionResponse**	**1.** The OCTT sends a **RequestStartTransactionRequest** with **idToken.idToken** *<Configured valid_idtoken_idtoken>* **idToken.type** *<Configured valid_idtoken_type>* **evseId** *<Configured evseId>*
	3. The Charging Station sends an **AuthorizeRequest** Note(s): *- This step needs to be executed when **AuthCtrlr.AuthorizeRemoteStart** is true, unless **AuthEnabled** is implemented with mutability ReadOnly AND the value is set to false.*	**4.** The OCTT responds with an **AuthorizeResponse** with: **idTokenInfo.status** *Accepted*
	5. The Charging Station sends a **TransactionEventRequest**	**6.** The OCTT responds with a **TransactionEventResponse**
Tool validations	* Step 2: Message: **RequestStartTransactionResponse** - **status** must be *Accepted* * Step 3: Message: **AuthorizeRequest** - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>* * Step 5: Message: **TransactionEventRequest** - **eventType** must be *Started* - **triggerReason** must be *RemoteStart* - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>* - **transactionInfo.remoteStartId** must be present	
	Post scenario validations: N/a	

Table 132. Test Case Id: TC_E_11_CS

Test case name	**Start transaction options - DataSigned**	
Test case Id	TC_E_11_CS	
Use case Id(s)	E01(S4)	
Requirement(s)	E01.FR.04, E01.FR.07, E01.FR.10, E01.FR.15, E01.FR.16	
System under test	Charging Station	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the start options.	
Purpose	To verify if the Charging Station starts a transaction when the EV and EVSE are connected and it has been configured to do so.	
Prerequisite(s)	- The Charging Station does NOT have the following configuration; The mutability of **TxStartPoint** is *ReadOnly* AND (the value *DataSigned* is NOT set OR (*ParkingBayOccupancy* OR *EVConnected* OR *Authorized*), is set). - If the mutability of **TxStartPoint** is *ReadWrite*, then the value *DataSigned* must be supported.	
Before (Preparations)	**Configuration State:** If the mutability of **TxStartPoint** is *ReadWrite* then **TxStartPoint** contains *DataSigned* **SampledDataCtrlr.SignReadings** is *true*	
	Memory State: N/a	
	Reusable State(s): **State is** *Authorized*	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Connect the EV and EVSE.*	
	1. The Charging Station notifies the CSMS about the status change of the connector.	**2.** The OCTT responds accordingly.
	3. The Charging Station sends a **TransactionEventRequest**	**4.** The OCTT responds with a **TransactionEventResponse**
	5. The Charging Station sends a **TransactionEventRequest**	**6.** The OCTT responds with a **TransactionEventResponse**

Test case name	**Start transaction options - DataSigned**
Tool validations	* Step 1: Message: **StatusNotificationRequest** - **connectorStatus** must be *Occupied* Message: **NotifyEventRequest** - **eventData[0].trigger** must be *Delta* - **eventData[0].actualValue** must be *Occupied* - **eventData[0].component.name** must be *Connector* - **eventData[0].variable.name** must be *AvailabilityState* * Step 3: Message: **TransactionEventRequest** - **eventType** must be *Started* - If the OCTT is configured to start transactions using a RequestStartTransactionRequest message then **triggerReason** must be *RemoteStart* or *SignedDataReceived* Else **triggerReason** must be *SignedDataReceived* - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>* - **evse** must be provided - **evse.connectorId** must be provided - **meterValue** is provided with the following values: **sampledValue.context** is *Transaction.Begin* **sampledValue.signedMeterValue.encodingMethod** is not omitted **sampledValue.signedMeterValue.publicKey** is not omitted **sampledValue.signedMeterValue.signedMeterData** is not omitted **sampledValue.signedMeterValue.signingMethod** is not omitted * Step 5: Message: **TransactionEventRequest** - **eventType** must be *Updated* - **triggerReason** must be *ChargingStateChanged* - **transactionInfo.chargingState** must be *Charging*
	Post scenario validations: N/a

Table 133. Test Case Id: TC_E_12_CS

Test case name	**Start transaction options - ParkingBayOccupied**	
Test case Id	TC_E_12_CS	
Use case Id(s)	E01(S1)	
Requirement(s)	E01.FR.01, E01.FR.07, E01.FR.10, E01.FR.15, E01.FR.16	
System under test	Charging Station	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the start options.	
Purpose	To verify if the Charging Station starts a transaction when the EV and EVSE are connected and it has been configured to do so.	
Prerequisite(s)	- The Charging Station does NOT have the following configuration; The mutability of **TxStartPoint** is *ReadOnly* AND the value *ParkingBayOccupancy* is NOT set. - If the mutability of **TxStartPoint** is *ReadWrite*, then the value *ParkingBayOccupancy* must be supported.	
Before (Preparations)	**Configuration State:** If the mutability of **TxStartPoint** is *ReadWrite* then **TxStartPoint** contains *ParkingBayOccupancy*	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Scenario)	**Charging Station**	**CSMS**
	Manual Action: *Drive EV into parking bay.*	
	1. The Charging Station sends a **TransactionEventRequest**	**2.** The OCTT responds with a **TransactionEventResponse**
Tool validations	* Step 1: Message: **TransactionEventRequest** - **eventType** must be *Started* - **triggerReason** must be *EVDetected*	
	Post scenario validations: N/a	

Table 134. Test Case Id: TC_E_16_CS

Test case name	**Stop transaction options - Deauthorized - Invalid idToken**	
Test case Id	TC_E_16_CS	
Use case Id(s)	E06(S3)	
Requirement(s)	E06.FR.04, E06.FR.15 & C15.FR.04	
System under test	Charging Station	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the start options.	
Purpose	To verify if the Charging Station stops a transaction when the transaction gets deauthorized by the status from the idTokenInfo at a TransactionEventResponse message and it has been configured to do so.	
Prerequisite(s)	- The Charging Station does NOT have the following configuration; The mutability of **TxStopPoint** is *ReadOnly* AND the value *Authorized* OR *PowerPathClosed* is NOT set OR (*EnergyTransfer* OR *DataSigned* is set). - If the mutability of **TxStopPoint** is *ReadWrite*, then the value *Authorized* OR *PowerPathClosed* must be supported. - The Charging Station supports local start/stop transaction. - The Charging Station supports authorization methods other than NoAuthorization	
Before (Preparations)	**Configuration State:** If the mutability of **TxStopPoint** is *ReadWrite* then **TxStopPoint** contains *PowerPathClosed* AND/OR *Authorized* **AuthCtrlr.AuthEnabled** is *true* (If implemented AND ReadWrite) **AuthCtrlr.DisableRemoteAuthorization** is *false* (If implemented) **OfflineTxForUnknownIdEnabled** is *true* (If implemented) **StopTxOnInvalidId** is *true*	
	Memory State: *IdTokenCached* for <*Configured valid idtoken fields*> (If implemented) *IdTokenLocalAuthList* for <*Configured valid idtoken fields*> (If implemented)	
	Reusable State(s): **State is** *StartOfflineTransaction*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The Charging Stations sends a **TransactionEventRequest** Note(s): *- The Charging Station will empty its Transaction message queue. This will contain one or more TransactionEventRequest messages*	**2.** The OCTT responds with a **TransactionEventResponse** Note(s): *- The OCTT will respond to the TransactionEventRequest containing the idToken, with* ***idtokenInfo.status*** *Invalid*
	3. The Charging Stations sends a **TransactionEventRequest** Note(s): *- After having emptied its queue, the Charging Station will send a TransactionEventRequest in which it reports it deauthorizes the transaction.*	**4.** The OCTT responds with a **TransactionEventResponse**
Tool validations	* Step 1: Message: **TransactionEventRequest** - **offline** must be *true* * Step 3: Message: **TransactionEventRequest** - **eventType** must be *Ended* - **triggerReason** must be *Deauthorized* - **transactionInfo.stoppedReason** is *DeAuthorized*	
	Post scenario validations: N/a	

Table 135. Test Case Id: TC_E_17_CS

Test case name	**Stop transaction options - Deauthorized - EV side disconnect**	
Test case Id	TC_E_17_CS	
Use case Id(s)	E06(S3)	
Requirement(s)	E06.FR.04, E06.FR.15	
System under test	Charging Station	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the start options.	
Purpose	To verify if the Charging Station stops a transaction when the transaction gets deauthorized by a connection loss from the EV side and it has been configured to do so.	
Prerequisite(s)	- The Charging Station does NOT have the following configuration; The mutability of **TxStopPoint** is *ReadOnly* AND the value *Authorized* OR *PowerPathClosed* is NOT set OR (*EnergyTransfer* OR *DataSigned* OR *EVConnected is set).* - *If the mutability of **TxStopPoint** is _ReadWrite*, then the value *Authorized* OR *PowerPathClosed* must be supported.	
Before (Preparations)	**Configuration State:** If the mutability of **TxStopPoint** is *ReadWrite* then **TxStopPoint** contains *PowerPathClosed* AND/OR *Authorized* **StopTxOnEVSideDisconnect** is *true* **UnlockOnEVSideDisconnect** is *false* **AuthCtrlr.AuthEnabled** is *true* (If implemented AND ReadWrite) **AuthCtrlr.DisableRemoteAuthorization** is *false* (If implemented)	
	Memory State: N/a	
	Reusable State(s): **State is** *EnergyTransferSuspended*	
Main (Scenario)	**Charging Station**	**CSMS**
	Manual Action: *Disconnect the EV and EVSE on EV side (EVSE loses connection with EV).*	
	1. The Charging Station sends a **TransactionEventRequest**	**2.** The OCTT responds with a **TransactionEventResponse**
	Manual Action: *Present the IdToken that was used to start the transaction.* Note(s): *- This manual action needs to be executed when the Charging Station has a detachable cable on the Charging Station side AND UnlockOnEVSideDisconnect is set to false.*	
	Manual Action: *Disconnect the EV and EVSE on Charging Station side.* Note(s): *- This manual action needs to be executed when the Charging Station has a detachable cable on the Charging Station side.*	
	3. The Charging Station notifies the CSMS about the current state of the connector.	**4.** The OCTT responds accordingly.

Test case name	**Stop transaction options - Deauthorized - EV side disconnect**
Tool validations	* Step 1: Message: **TransactionEventRequest** - **triggerReason** must be *EVCommunicationLost* - **transactionInfo.chargingState** must be *Idle* - **transactionInfo.stoppedReason** must be *EVDisconnected* - **eventType** must be *Ended* * Step 3: Message: **StatusNotificationRequest** - **connectorStatus** *Available* Message: **NotifyEventRequest** - **eventData[0].trigger** *Delta* - **eventData[0].actualValue** *"Available"* - **eventData[0].component.name** *"Connector"* - **eventData[0].variable.name** *"AvailabilityState"*
	Post scenario validations: N/a

Table 136. Test Case Id: TC_E_39_CS

Test case name	**Stop transaction options - Deauthorized - timeout**	
Test case Id	TC_E_39_CS	
Use case Id(s)	E06(S3)	
Requirement(s)	E06.FR.04, E06.FR.15	
System under test	Charging Station	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the start options.	
Purpose	To verify if the Charging Station stops a transaction when the transaction gets deauthorized because the cable was not plugged in within the Configured durationout and it has been configured to do so.	
Prerequisite(s)	- The Charging Station does NOT have the following configuration; The mutability of **TxStopPoint** is *ReadOnly* AND the value *Authorized* is NOT set. - If the mutability of **TxStopPoint** is *ReadWrite*, then the value *Authorized* must be supported.	
Before (Preparations)	**Configuration State:** If the mutability of **TxStopPoint** is *ReadWrite* then **TxStopPoint** contains *Authorized* - **TxCtrlr.EVConnectionTimeOut** is *<Configured ev_connection_timeout>* - **AuthCtrlr.AuthEnabled** is *true* (If implemented AND ReadWrite) **AuthCtrlr.DisableRemoteAuthorization** is *false* (If implemented)	
	Memory State: N/a	
	Reusable State(s): **State** is *Authorized*	
Main (Test scenario)	**Charging Station**	**CSMS**
	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.*	

Test case name	**Stop transaction options - Deauthorized - timeout**
Tool validations	* Step 1: Message: **TransactionEventRequest** - **triggerReason** must be *EVConnectTimeout* - **eventType** must be *Updated* if TxStartPoint is *ParkingBayOccupancy*, else *Ended* - **transactionInfo.stoppedReason** must be *Timeout* * Step 3: Message: **TransactionEventRequest** - **triggerReason** can only be *CablePluggedIn* - **transactionInfo.chagringState** should not be *Charging* - **eventType** must be *Updated* if TxStartPoint is *ParkingBayOccupancy*, else *Ended*
	Post scenario validations: N/a

Table 137. Test Case Id: TC_E_03_CS

Test case name	**Local start transaction - Cable plugin first - Success**	
Test case Id	TC_E_03_CS	
Use case Id(s)	E02 AND (C01 OR C02 OR C04 OR C06)	
Requirement(s)	E02.FR.01, E02.FR.05, E02.FR.06, E02.FR.07, E02.FR.13, E02.FR.15, E02.FR.16, E02.FR.17, E01.FR.16 AND C01.FR.02, C02.FR.01, C06.FR.02	
System under test	Charging Station	
Description	OCPP 2.x.x allows an EV driver to either first connect the EV and EVSE OR present a form of identification. Both sequences will result in being able to charge.	
Purpose	To verify if the Charging Station is able to start a charging session when the EV driver first connects the EV and EVSE, before authorization.	
Prerequisite(s)	- The Charging Station supports at least one authorization method described at the following Use cases; C01, C02, C04, C06. - The Charging Station does NOT have a cable lock, which prevents the EV driver to connect the EV and EVSE before authorization.	
Before (Preparations)	**Configuration State:** **AuthCtrlr.AuthEnabled** is *true* (If implemented AND ReadWrite) **AuthCtrlr.DisableRemoteAuthorization** is *false* (If implemented)	
	Memory State: N/a	
	Reusable State(s): **State is** *EVConnectedPreSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *Authorized* (local)	
	2. Execute **Reusable State** *EnergyTransferStarted*	
Tool validations	N/a	
	Post scenario validations: N/a	

Table 138. Test Case Id: TC_E_04_CS

Test case name	**Local start transaction - Authorization first - Success**	
Test case Id	TC_E_04_CS	
Use case Id(s)	E03 AND (C01 OR C02 OR C04 OR C06)	
Requirement(s)	E03.FR.01, E03.FR.06, E03.FR.12, E01.FR.16 AND C01.FR.02, C02.FR.01, C06.FR.02	
System under test	Charging Station	
Description	OCPP 2.x.x allows an EV driver to either first connect the EV and EVSE OR present a form of identification. Both sequences will result in being able to charge.	
Purpose	To verify if the Charging Station is able to start a charging session when the EV driver first presends a form of identification, before connecting the EV and EVSE.	
Prerequisite(s)	The Charging Station supports at least one authorization method described at the following Use cases; C01, C02, C04, C06.	
Before (Preparations)	**Configuration State:** **AuthCtrlr.AuthEnabled** is *true* (If implemented AND ReadWrite) **AuthCtrlr.DisableRemoteAuthorization** is *false* (If implemented)	
	Memory State: N/a	
	Reusable State(s): **State is** *ParkingBayOccupied* (Optional state)	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *Authorized* (local)	
	2. Execute **Reusable State** *EnergyTransferStarted*	
Tool validations	N/a	
	Post scenario validations: N/a	

Table 139. Test Case Id: TC_E_05_CS

Test case name	**Local start transaction - Authorization first - Cable plugin timeout**	
Test case Id	TC_E_05_CS	
Use case Id(s)	E03 AND (C01 OR C02 OR C04 OR C06)	
Requirement(s)	E03.FR.01, E03.FR.05, E03.FR.06, E03.FR.12 AND C01.FR.02, C02.FR.01, C06.FR.02	
System under test	Charging Station	
Description	OCPP 2.x.x allows an EV driver to either first connect the EV and EVSE OR present a form of identification. Both sequences will result in being able to charge.	
Purpose	To verify if the Charging Station is able to deauthorize the transaction after the **EVConnectionTimeout** has expired.	
Prerequisite(s)	The Charging Station supports at least one authorization method described at the following Use cases; C01, C02, C04, C06.	
Before (Preparations)	**Configuration State:** - **TxCtrlr.EVConnectionTimeOut** is *<Configured ev_connection_timeout>* - **AuthCtrlr.AuthEnabled** is *true* (If implemented AND ReadWrite) - **AuthCtrlr.DisableRemoteAuthorization** is *false* (If implemented) - **AuthCacheCtrlr.Enabled** is *false* (If implemented) - **AuthCtrlr.LocalPreAuthorize** is *false*	
	Memory State: N/a	
	Reusable State(s): **State is** *Authorized* (local)	
Main (Test scenario)	**Charging Station**	**CSMS**
	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): *- This step is only executed if TxStartPoint is ParkingBayOccupancy or Authorized* *- Optionally the Charging Station can send a StatusNotificationRequest or NotifyEventRequest with status Available*	
	3. Execute **Reusable State** *Authorized* (local) Note(s): *- This step is executed to verify if the EVSE is actually ready to start another charging session.*	
	4. Execute **Reusable State** *EnergyTransferStarted*	
Tool validations	* Step 1: Message: **TransactionEventRequest** - **triggerReason** must be *EVConnectTimeout* If <Configured TxStopPoint> contains *Authorized* then **eventType** must be *Ended* AND **transactionInfo.stoppedReason** must be *Timeout* Else **eventType** must be *Updated*	
	Post scenario validations: N/a	

Table 140. Test Case Id: TC_E_38_CS

Test case name	**Local start transaction - EV not ready**	
Test case Id	TC_E_38_CS	
Use case Id(s)	E03	
Requirement(s)	N/a	
System under test	Charging Station	
Description	OCPP 2.x.x allows an EV driver to either first connect the EV and EVSE OR present a form of identification. Both sequences will result in being able to charge.	
Purpose	To verify if the Charging Station is able to handle and report if an EV is not ready to start the energy transfer (yet).	
Prerequisite(s)	TxStartPoint should not be EnergyTransfer	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): **State is** *Authorized*	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Set the EV to a state in which it is NOT ready for energy transfer.*	
	1. Execute **Reusable State** *EVConnectedPreSession*	
	2. The Charging Station sends a **TransactionEventRequest**	**3.** The OCTT responds with a **TransactionEventResponse**
Tool validations	* Step 2: Message: **TransactionEventRequest** - **triggerReason** must be *ChargingStateChanged* - **transactionInfo.chargingState** must be *SuspendedEV*	

Table 141. Test Case Id: TC_E_52_CS

Test case name	**Local start transaction - Authorization first - DisableRemoteAuthorization**	
Test case Id	TC_E_52_CS	
Use case Id(s)	E03 AND C01	
Requirement(s)	C01.FR.02, C01.FR.05,	
System under test	Charging Station	
Description	When DisableRemoteAuthorization is set to true, the Charging Station will only try to look up an IdToken in Authorization Cache or Local Authorization List, and not do an AuthorizeRequest for IdTokens. This overrules requirement C01.FR.02 and C01.FR.05.	
Purpose	To verify that the Charging Station will not send an AuthorizeRequest when DisableRemoteAuthorization is set to true.	
Prerequisite(s)	The Charging Station supports the authorization method described in C01. (RFID) AuthCtrlr.DisableRemoteAuthorization is implemented.	
Before (Preparations)	**Configuration State:** **AuthCtrlr.Enabled** is *true* (If implemented) **AuthCtrlr.DisableRemoteAuthorization** is *true*	
	Memory State: None of the configured valid IdTokens is present in Authorization Cache or Local Authorization List.	
	Reusable State(s): **State is** *ParkingBayOccupied* (Optional state)	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Present an idToken which is not configured in the Local Authorization List nor present in Authorization Cache.*	
	1. The Charging Station does NOT send a **AuthorizeRequest**	
Tool validations	* Step 1: Check that Charging Station does NOT send an AuthorizeRequest and authorization is refused.	
	Post scenario validations: - N/a	

Table 142. Test Case Id: TC_E_06_CS

Test case name	**Local Stop Transaction - Accepted**	
Test case Id	TC_E_06_CS	
Use case Id(s)	E07 AND (C01 OR C02 OR C04)	
Requirement(s)	E07.FR.04, E06.FR.15 AND C01.FR.03	
System under test	Charging Station	
Description	The EV Driver is able to stop an ongoing transaction, by locally presenting an IdToken.	
Purpose	To verify whether the Charging Station is able to perform a local stop authorization.	
Prerequisite(s)	The Charging Station supports at least one authorization method described at the following Use cases; C01, C02, C04.	
Before (Preparations)	**Configuration State:** **AuthCtrlr.AuthEnabled** is *true* (If implemented AND ReadWrite) **AuthCtrlr.DisableRemoteAuthorization** is *false* (If implemented)	
	Memory State: N/a	
	Reusable State(s): **State is** *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *StopAuthorized* (local)	
	2. Execute **Reusable State** *EVConnectedPostSession*	
	3. Execute **Reusable State** *EVDisconnected*	
	4. Execute **Reusable State** *ParkingBayUnoccupied*	
Tool validations	N/a	
	Post scenario validations: N/a	

Table 143. Test Case Id: TC_E_07_CS

Test case name	**Stop transaction options - PowerPathClosed - Local stop**	
Test case Id	TC_E_07_CS	
Use case Id(s)	E06(S5)	
Requirement(s)	E06.FR.06, E06.FR.15	
System under test	Charging Station	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the stop options.	
Purpose	To verify if the Charging Station stops a transaction when it is locally stopped by an EV driver and TxStopPoint contains *PowerPathClosed*.	
Prerequisite(s)	- The Charging Station does NOT have the following configuration; The mutability of **TxStopPoint** is *ReadOnly* AND (the value *PowerPathClosed* is NOT set OR *Authorized* is set). - If the mutability of **TxStopPoint** is *ReadWrite*, then the value *PowerPathClosed* must be supported.	
Before (Preparations)	**Configuration State:** **TxStopPoint** contains *PowerPathClosed*	
	Memory State: N/a	
	Reusable State(s): **State is** *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Present IdToken to stop charging session.*	
	1. Execute **Reusable State** *StopAuthorized*	
Tool validations	N/a	
	Post scenario validations: N/a	

Table 144. Test Case Id: TC_E_35_CS

Test case name	**Stop transaction options - PowerPathClosed - Remote stop**	
Test case Id	TC_E_35_CS	
Use case Id(s)	E06(S5)	
Requirement(s)	E06.FR.06, E06.FR.15	
System under test	Charging Station	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the stop options.	
Purpose	To verify if the Charging Station stops a transaction when it is remotely stopped the CSMS and TxStopPoint contains *PowerPathClosed*.	
Prerequisite(s)	- The Charging Station does NOT have the following configuration; The mutability of **TxStopPoint** is *ReadOnly* AND (the value *PowerPathClosed* is NOT set OR *Authorized* is set). - If the mutability of **TxStopPoint** is *ReadWrite*, then the value *PowerPathClosed* must be supported.	
Before (Preparations)	**Configuration State:** **TxStopPoint** contains *PowerPathClosed*	
	Memory State: N/a	
	Reusable State(s): **State is** *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **RequestStopTransactionResponse**	**1.** The OCTT sends a **RequestStopTransactionRequest** with **transactionId** *<transactionId provided by the Charging Station in **TransactionEventRequest**>*
	3. The Charging Station sends a **TransactionEventRequest**	**4.** The OCTT responds with a **TransactionEventResponse**
Tool validations	* Step 2: Message: **RequestStopTransactionResponse** - **status** must be *Accepted* * Step 3: Message: **TransactionEventRequest** - **triggerReason** must be *RemoteStop* - **transactionInfo.stoppedReason** must be *Remote* - **eventType** must be *Ended*	
	Post scenario validations: N/a	

Table 145. Test Case Id: TC_E_37_CS

Test case name	**Stop transaction options - PowerPathClosed - EV side disconnect**	
Test case Id	TC_E_37_CS	
Use case Id(s)	E06(S5)	
Requirement(s)	E06.FR.06, E06.FR.15	
System under test	Charging Station	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the stop options.	
Purpose	To verify if the Charging Station stops a transaction when the EV and the EVSE get disconnected and TxStopPoint contains *PowerPathClosed*.	
Prerequisite(s)	- The Charging Station does NOT have the following configuration; The mutability of **TxStopPoint** is *ReadOnly* AND (the value *PowerPathClosed* is NOT set OR (*EnergyTransfer* OR *EVConnected* OR *DataSigned* is set)). - If the mutability of **TxStopPoint** is *ReadWrite*, then the value *PowerPathClosed* must be supported.	
Before (Preparations)	**Configuration State:** **TxStopPoint** contains *PowerPathClosed* **StopTxOnEVSideDisconnect** is *false* (If mutability is ReadWrite)	
	Memory State: N/a	
	Reusable State(s): **State is** *EnergyTransferSuspended*	
Main (Scenario)	**Charging Station**	**CSMS**
	Manual Action: *Disconnect the EV and EVSE on EV side (EVSE loses connection with EV).*	
	1. The Charging Station sends a **TransactionEventRequest**	**2.** The OCTT responds with a **TransactionEventResponse**
Tool validations	* Step 1: Message: **TransactionEventRequest** - **triggerReason** must be *EVCommunicationLost* - **transactionInfo.chargingState** must be *Idle* - **transactionInfo.stoppedReason** must be *EVDisconnected* or *StoppedByEV* (preferred value) - **eventType** must be *Ended*	
	Post scenario validations: N/a	

Table 146. Test Case Id: TC_E_08_CS

Test case name	**Stop transaction options - EnergyTransfer stopped - StopAuthorized**	
Test case Id	TC_E_08_CS	
Use case Id(s)	E06(S6)	
Requirement(s)	E06.FR.07, E06.FR.15	
System under test	Charging Station	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the stop options.	
Purpose	To verify if the Charging Station stops a transaction when the energy transfer stopped normally and it has been configured to do so.	
Prerequisite(s)	- The Charging Station does NOT have the following configuration; The mutability of **TxStopPoint** is *ReadOnly* AND (the value *EnergyTransfer* is NOT set OR (*Authorized* OR *PowerPathClosed*) is set). - If the mutability of **TxStopPoint** is *ReadWrite*, then the value *EnergyTransfer* must be supported.	
Before (Preparations)	**Configuration State:** **TxStopPoint** contains *EnergyTransfer*	
	Memory State: N/a	
	Reusable State(s): **State is** *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. State is *StopAuthorized*	
Tool validations	* Step 1: N/a	
	Post scenario validations: N/a	

Table 147. Test Case Id: TC_E_22_CS

Test case name	**Stop transaction options - EnergyTransfer stopped - SuspendedEV**	
Test case Id	TC_E_22_CS	
Use case Id(s)	E06(S6)	
Requirement(s)	E06.FR.07, E06.FR.15	
System under test	Charging Station	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the stop options.	
Purpose	To verify if the Charging Station stops a transaction when the energy transfer stopped by the EV and the Charging Station has been configured to do so.	
Prerequisite(s)	- The Charging Station does NOT have the following configuration; The mutability of **TxStopPoint** is *ReadOnly* AND the value *EnergyTransfer* is NOT set. - If the mutability of **TxStopPoint** is *ReadWrite*, then the value *EnergyTransfer* must be supported.	
Before (Preparations)	**Configuration State:** **TxStopPoint** contains *EnergyTransfer*	
	Memory State: N/a	
	Reusable State(s): **State is** *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	<u>Manual Action</u>: *The EV suspends the energy transfer.*	
	1. The Charging Station sends a **TransactionEventRequest**	**2.** The OCTT responds with a **TransactionEventResponse**
Tool validations	* 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* (if chargingState is *EVConnected*) OR - **eventType** must be *Updated* (if chargingState is *SuspendedEV*)	
	Post scenario validations: N/a	

Table 148. Test Case Id: TC_E_14_CS

Test case name	**Stop transaction options - EVDisconnected - Charging Station side**	
Test case Id	TC_E_14_CS	
Use case Id(s)	E06(S2)	
Requirement(s)	E06.FR.02, E06.FR.15	
System under test	Charging Station	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the stop options.	
Purpose	To verify if the Charging Station stops a transaction when the EV and EVSE are disconnected at the Charging Station side and it has been configured to do so.	
Prerequisite(s)	- The Charging Station does NOT have the following configuration; The mutability of **TxStopPoint** is *ReadOnly* AND (the value *EVConnected* is NOT set OR (*EnergyTransfer* OR *PowerPathClosed* OR *DataSigned* OR *Authorized* is set)). - If the mutability of **TxStopPoint** is *ReadWrite*, then the value *EVConnected* must be supported.	
Before (Preparations)	**Configuration State:** **TxStopPoint** contains *EVConnected*	
	Memory State: N/a	
	Reusable State(s): **State is** *EVConnectedPostSession*	
Main (Scenario)	**Charging Station**	**CSMS**
	Manual Action: *Disconnect the EV and EVSE.*	
	1. The Charging Station notifies the CSMS about the status change of the connector.	**2.** The OCTT responds accordingly.
	3. The Charging Station sends a **TransactionEventRequest**	**4.** The OCTT responds with a **TransactionEventResponse**
Tool validations	* Step 1: Message: **StatusNotificationRequest** - **connectorStatus** must be *Available* Message: **NotifyEventRequest** - **eventData[0].trigger** must be *Delta* - **eventData[0].actualValue** must be *Available* - **eventData[0].component.name** must be *Connector* - **eventData[0].variable.name** must be *AvailabilityState* * 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*	
	Post scenario validations: N/a	

Table 149. Test Case Id: TC_E_20_CS

Test case name	**Stop transaction options - EVDisconnected - EV side (able to charge IEC 61851-1 EV)**	
Test case Id	TC_E_20_CS	
Use case Id(s)	E06(S2), E10	
Requirement(s)	E06.FR.02, E06.FR.15	
System under test	Charging Station	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the stop options.	
Purpose	To verify if the Charging Station stops a transaction when the EV and EVSE are disconnected at the EV side and it has been configured to do so.	
Prerequisite(s)	- The Charging Station does NOT have the following configuration; The mutability of **TxStopPoint** is *ReadOnly* AND (the value *EVConnected* is NOT set OR (*EnergyTransfer* OR *PowerPathClosed* is set)). - If the mutability of **TxStopPoint** is *ReadWrite*, then the value *EVConnected* must be supported. - The Charging Station does NOT have following configuration combination; **StopTxOnEVSideDisconnect** mutability ReadOnly with value *true* AND TxStopPoint mutability is *ReadOnly* and contains *Authorized*. - The Charging Station is able to charge with a EV that uses IEC 61851-1.	
Before (Preparations)	**Configuration State:** **TxStopPoint** contains *EVConnected* **StopTxOnEVSideDisconnect** is *false* (If mutability is ReadWrite)	
	Memory State: N/a	
	Reusable State(s): **State is** *EnergyTransferSuspended*	
Main (Scenario)	**Charging Station**	**CSMS**
	Manual Action: *Disconnect the EV and EVSE on EV side (EVSE loses connection with EV).*	
	1. The Charging Station sends a **TransactionEventRequest**	**2.** The OCTT responds with a **TransactionEventResponse**
Tool validations	* Step 1: Message: **TransactionEventRequest** - **triggerReason** must be *EVCommunicationLost* - **transactionInfo.chargingState** must be *Idle* - **transactionInfo.stoppedReason** must be *EVDisconnected* - **eventType** must be *Ended*	
	Post scenario validations: N/a	

Table 150. Test Case Id: TC_E_54_CS

Test case name	**Stop transaction options - EVDisconnected - EV side (not able to charge IEC 61851-1 EV)**	
Test case Id	TC_E_54_CS	
Use case Id(s)	E06(S2), E10	
Requirement(s)	E06.FR.02, E06.FR.15	
System under test	Charging Station	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the stop options.	
Purpose	To verify if the Charging Station stops a transaction when the EV and EVSE are disconnected at the EV side and it has been configured to do so.	
Prerequisite(s)	- The Charging Station does NOT have the following configuration; The mutability of **TxStopPoint** is *ReadOnly* AND (the value *EVConnected* is NOT set OR (*EnergyTransfer* OR *PowerPathClosed* is set)). - If the mutability of **TxStopPoint** is *ReadWrite*, then the value *EVConnected* must be supported. - The Charging Station does NOT have following configuration combination; **StopTxOnEVSideDisconnect** mutability ReadOnly with value *true* AND TxStopPoint mutability is *ReadOnly* and contains *Authorized*. - The Charging Station supports high level communication.	
Before (Preparations)	**Configuration State:** **TxStopPoint** contains *EVConnected* **StopTxOnEVSideDisconnect** is *false* (If mutability is ReadWrite)	
	Memory State: N/a	
	Reusable State(s): **State is** *EnergyTransferSuspended*	
Main (Scenario)	**Charging Station**	**CSMS**
	Manual Action: *Disconnect the EV and EVSE on EV side (EVSE loses connection with EV).*	
	1. The Charging Station sends a **TransactionEventRequest**	**2.** The OCTT responds with a **TransactionEventResponse**
Tool validations	* Step 1: Message: **TransactionEventRequest** - **triggerReason** must be *EVCommunicationLost* - **transactionInfo.chargingState** must be *Idle* - **transactionInfo.stoppedReason** must be *StoppedByEV* or *EVDisconnected* - **eventType** must be *Ended*	
	Post scenario validations: N/a	

Table 151. Test Case Id: TC_E_15_CS

Test case name	**Stop transaction options - StopAuthorized - Local**	
Test case Id	TC_E_15_CS	
Use case Id(s)	E06(S3)	
Requirement(s)	E06.FR.03, E06.FR.15	
System under test	Charging Station	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the stop options.	
Purpose	To verify if the Charging Station stops a transaction when the EV driver locally stops the transaction and it has been configured to do so.	
Prerequisite(s)	- The Charging Station supports at least one authorization method described at the following Use cases; C01, C02, C04. - The Charging Station does NOT have the following configuration; The mutability of **TxStopPoint** is *ReadOnly* AND the value *Authorized* is NOT set OR *PowerPathClosed* is set. - If the mutability of **TxStopPoint** is *ReadWrite*, then the value *Authorized* must be supported.	
Before (Preparations)	**Configuration State:** **TxStopPoint** contains *Authorized*	
	Memory State: N/a	
	Reusable State(s): **State is** *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	Notes(s): *The tool will wait for <Configured Transaction Duration> seconds*	
	Manual Action: *Present IdToken to stop charging session.*	
	1. The Charging Station sends a **TransactionEventRequest**	**2.** The OCTT responds with a **TransactionEventResponse**
Tool validations	* Step 1: Message: **TransactionEventRequest** - **triggerReason** must be *StopAuthorized* - **transactionInfo.stoppedReason** must be *Local* - **eventType** must be *Ended*	
	Post scenario validations: N/a	

Table 152. Test Case Id: TC_E_21_CS

Test case name	**Stop transaction options - StopAuthorized - Remote**	
Test case Id	TC_E_21_CS	
Use case Id(s)	E06(S3) AND F03	
Requirement(s)	E06.FR.03, E06.FR.15 AND F03.FR.09	
System under test	Charging Station	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the stop options.	
Purpose	To verify if the Charging Station stops a transaction when it receives a RequestStopTransactionRequest and it has been configured to do so.	
Prerequisite(s)	- The Charging Station does NOT have the following configuration; The mutability of **TxStopPoint** is *ReadOnly* AND the value *Authorized* is NOT set OR *PowerPathClosed* is set. - If the mutability of **TxStopPoint** is *ReadWrite*, then the value *Authorized* must be supported.	
Before (Preparations)	**Configuration State:** **TxStopPoint** contains *Authorized*	
	Memory State: N/a	
	Reusable State(s): **State is** *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **RequestStopTransactionResponse**	**1.** The OCTT sends a **RequestStopTransactionRequest** with **transactionId** *<transactionId provided by the Charging Station in* ***TransactionEventRequest****>*
	3. The Charging Station sends a **TransactionEventRequest**	**4.** The OCTT responds with a **TransactionEventResponse**
Tool validations	* Step 2: Message: **RequestStopTransactionResponse** - **status** must be *Accepted* * Step 3: Message: **TransactionEventRequest** - **triggerReason** must be *RemoteStop* - **transactionInfo.stoppedReason** must be *Remote* - **eventType** must be *Ended*	
	Post scenario validations: N/a	

Table 153. Test Case Id: TC_E_19_CS

Test case name	**Stop transaction options - ParkingBayUnoccupied**	
Test case Id	TC_E_19_CS	
Use case Id(s)	E06(S1)	
Requirement(s)	E06.FR.01, E06.FR.15	
System under test	Charging Station	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the stop options.	
Purpose	To verify if the Charging Station stops a transaction when the EV left the parking bay and it has been configured to do so.	
Prerequisite(s)	- The Charging Station does NOT have the following configuration; The mutability of **TxStopPoint** is *ReadOnly* AND (the value *ParkingBayOccupied* is NOT set OR (*EnergyTransfer* OR *PowerPathClosed* OR *DataSigned* OR *Authorized* OR *EVConnected* is set)). - If the mutability of **TxStopPoint** is *ReadWrite*, then the value *ParkingBayOccupied* must be supported.	
Before (Preparations)	**Configuration State:** **TxStopPoint** contains *ParkingBayOccupied*	
	Memory State: N/a	
	Reusable State(s): **State is** *EVDisconnected*	
Main (Scenario)	**Charging Station**	**CSMS**
	Manual Action: *Drive EV out of parking bay.*	
	1. The Charging Station sends a **TransactionEventRequest**	**2.** The OCTT responds with a **TransactionEventResponse**
Tool validations	* Step 1: Message: **TransactionEventRequest** - **triggerReason** must be *EVDeparted* - If the OCTT is configured to stop transactions using a RequestStopTransactionRequest message then **transactionInfo.stoppedReason** must be *Remote* Else **transactionInfo.stoppedReason** must be *Local* - **eventType** must be *Ended*	
	Post scenario validations: N/a	

Table 154. Test Case Id: TC_E_24_CS

Test case name	**Disconnect cable on EV-side - Deauthorize transaction - UnlockOnEVSideDisconnect is true**	
Test case Id	TC_E_24_CS	
Use case Id(s)	E09	
Requirement(s)	E09.FR.01, E09.FR.02, E09.FR.04	
System under test	Charging Station	
Description	The Charging Station can behave in several different ways when the cable is disconnected at the EV side, based on its configuration. This test case tests one of the possible configuration settings.	
Purpose	To verify if the Charging Station deauthorizes the transaction when the EV and EVSE are disconnected at the EV side and it has been configured to do so AND unlocks the cable at Charging Station side.	
Prerequisite(s)	The Charging Station does NOT have a permanently attached cable.	
Before (Preparations)	**Configuration State:** **StopTxOnEVSideDisconnect** is *true* **UnlockOnEVSideDisconnect** is *true*	
	Memory State: N/a	
	Reusable State(s): **State is** *EnergyTransferSuspended*	
Main (Scenario)	**Charging Station**	**CSMS**
	Manual Action: *Disconnect the EV and EVSE on EV side (EVSE loses connection with EV).*	
	1. The Charging Station sends a **TransactionEventRequest**	**2.** The OCTT responds with a **TransactionEventResponse**
	Manual Action: *Disconnect the EV and EVSE on Charging Station side.*	
	3. The Charging Station notifies the CSMS about the current state of the connector.	**4.** The OCTT responds accordingly.
Tool validations	* Step 1: Message: **TransactionEventRequest** - **triggerReason** must be *EVCommunicationLost* - **transactionInfo.chargingState** must be *Idle* * Step 3: Message: **StatusNotificationRequest** - **connectorStatus** *Available* Message: **NotifyEventRequest** - **eventData[0].trigger** *Delta* - **eventData[0].actualValue** *"Available"* - **eventData[0].component.name** *"Connector"* - **eventData[0].variable.name** *"AvailabilityState"*	
	Post scenario validations: N/a	

Table 155. Test Case Id: TC_E_25_CS

Test case name	**Disconnect cable on EV-side - Deauthorize transaction - UnlockOnEVSideDisconnect is false**	
Test case Id	TC_E_25_CS	
Use case Id(s)	E09	
Requirement(s)	E09.FR.01, E09.FR.03, E09.FR.04	
System under test	Charging Station	
Description	The Charging Station can behave in several different ways when the cable is disconnected at the EV side, based on its configuration. This test case tests one of the possible configuration settings.	
Purpose	To verify if the Charging Station deauthorizes the transaction when the EV and EVSE are disconnected at the EV side and it has been configured to do so AND keeps the cable locked at Charging Station side.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** **StopTxOnEVSideDisconnect** is *true* **UnlockOnEVSideDisconnect** is *false*	
	Memory State: N/a	
	Reusable State(s): **State is** *EnergyTransferSuspended*	
Main (Scenario)	**Charging Station**	**CSMS**
	Manual Action: *Disconnect the EV and EVSE on EV side (EVSE loses connection with EV).*	
	1. The Charging Station sends a **TransactionEventRequest**	**2.** The OCTT responds with a **TransactionEventResponse**
	Manual Action: *Present the IdToken that was used to start the transaction.* Note(s): *- This manual action needs to be executed when the Charging Station has a detachable cable on the Charging Station side.*	
	Manual Action: *Disconnect the EV and EVSE on Charging Station side.* Note(s): *- This manual action needs to be executed when the Charging Station has a detachable cable on the Charging Station side.*	
	3. The Charging Station notifies the CSMS about the current state of the connector.	**4.** The OCTT responds accordingly.
Tool validations	* Step 1: Message: **TransactionEventRequest** - **triggerReason** must be *EVCommunicationLost* - **transactionInfo.chargingState** must be *Idle* * Step 3: Message: **StatusNotificationRequest** - **connectorStatus** *Available* Message: **NotifyEventRequest** - **eventData[0].trigger** *Delta* - **eventData[0].actualValue** *"Available"* - **eventData[0].component.name** *"Connector"* - **eventData[0].variable.name** *"AvailabilityState"*	
	Post scenario validations: N/a	

Table 156. Test Case Id: TC_E_26_CS

Test case name	**Disconnect cable on EV-side - Suspend transaction**	
Test case Id	TC_E_26_CS	
Use case Id(s)	E10	
Requirement(s)	E10.FR.01, E10.FR.03	
System under test	Charging Station	
Description	The Charging Station can behave in several different ways when the cable is disconnected at the EV side, based on its configuration. This test case tests one of the possible configuration settings.	
Purpose	To verify if the Charging Station suspends the transaction when the EV and EVSE are disconnected at the EV side and it has been configured to do so AND is able restart the energy transfer after reconnecting the EV and EVSE.	
Prerequisite(s)	- The Charging Station does NOT have the following configuration; The mutability of **TxStopPoint** is *ReadOnly* AND (the value *Authorized* OR *ParkingBayOccupancy* is NOT set OR (*EnergyTransfer* OR *PowerPathClosed* OR *DataSigned* OR *EVConnected* is set)). - If the mutability of **TxStopPoint** is *ReadWrite*, then the value *Authorized* OR *ParkingBayOccupancy* must be supported.	
Before (Preparations)	**Configuration State:** **TxStopPoint** contains *Authorized* (If supported) AND/OR *ParkingBayOccupancy* (If supported) **UnlockOnEVSideDisconnect** is *false* **StopTxOnEVSideDisconnect** is *false*	
	Memory State: N/a	
	Reusable State(s): **State is** *EnergyTransferSuspended*	
Main (Scenario)	**Charging Station**	**CSMS**
	Manual Action: *Disconnect the EV and EVSE on EV side (EVSE loses connection with EV).*	
	1. The Charging Station sends a **TransactionEventRequest**	**2.** The OCTT responds with a **TransactionEventResponse**
	3. The Charging Station notifies the CSMS about the current state of the connector. Note(s): *- This step needs to be executed when the Charging Station has a permanently attached cable on the Charging Station side.*	**4.** The OCTT responds accordingly.
	Manual Action: *Reconnect the EV and EVSE on EV side.* Note(s): *- If the Charging Station has a permanently attached cable on the Charging Station side, then this step needs to be executed before the configured EVConnectionTimeout expires.*	
	5. The Charging Station sends a **TransactionEventRequest**	**6.** The OCTT responds with a **TransactionEventResponse**
	7. The Charging Station sends a **TransactionEventRequest**	**8.** The OCTT responds with a **TransactionEventResponse**

Test case name	**Disconnect cable on EV-side - Suspend transaction**
Tool validations	* Step 1: Message: **TransactionEventRequest** - **triggerReason** must be *EVCommunicationLost* - **transactionInfo.chargingState** must be *Idle* - **eventType** must be *Updated* * Step 3: Message: **StatusNotificationRequest** - **connectorStatus** *Available* Message: **NotifyEventRequest** - **eventData[0].trigger** *Delta* - **eventData[0].actualValue** *"Available"* - **eventData[0].component.name** *"Connector"* - **eventData[0].variable.name** *"AvailabilityState"* * Step 5: Message: **TransactionEventRequest** - **triggerReason** must be *CablePluggedIn* - **transactionInfo.chargingState** must be *EVConnected* - **eventType** must be *Updated* * Step 7: Message: **TransactionEventRequest** - **triggerReason** must be *ChargingStateChanged* - **transactionInfo.chargingState** must be *Charging* - **eventType** must be *Updated*
	Post scenario validations: N/a

Table 157. Test Case Id: TC_E_27_CS

Test case name	**Disconnect cable on EV-side - Suspend transaction - Fixed cable connection timeout**	
Test case Id	TC_E_27_CS	
Use case Id(s)	E10	
Requirement(s)	E10.FR.02, E10.FR.03	
System under test	Charging Station	
Description	The Charging Station can behave in several different ways when the cable is disconnected at the EV side, based on its configuration. This test case tests one of the possible configuration settings.	
Purpose	To verify if the Charging Station suspends the transaction when the EV and EVSE are disconnected at the EV side and it has been configured to do so AND deauthorizes the transaction after the configured connection timeout expires.	
Prerequisite(s)	- The Charging Station does NOT have the following configuration; The mutability of **TxStopPoint** is *ReadOnly* AND (the value *Authorized* OR *ParkingBayOccupancy* is NOT set OR (*EnergyTransfer* OR *PowerPathClosed* OR *DataSigned* OR *EVConnected is set)).* - *If the mutability of* ***TxStopPoint*** *is _ReadWrite*, then the value *Authorized* OR *ParkingBayOccupancy* must be supported. - The Charging Station has a permanently attached cable at the Charging Station side. - **StopTxOnEVSideDisconnect** can be set to *false*.	
Before (Preparations)	**Configuration State:** **TxStopPoint** contains *Authorized* (If supported) **TxStopPoint** contains *ParkingBayOccupancy* (If supported) **UnlockOnEVSideDisconnect** is *false* **StopTxOnEVSideDisconnect** is *false*	
	Memory State: N/a	
	Reusable State(s): **State is** *EnergyTransferSuspended*	
Main (Scenario)	**Charging Station**	**CSMS**
	Manual Action: *Disconnect the EV and EVSE on EV side (EVSE loses connection with EV).*	
	1. The Charging Station sends a **TransactionEventRequest**	**2.** The OCTT responds with a **TransactionEventResponse**
	3. The Charging Station notifies the CSMS about the current state of the connector. Note(s): *- This step needs to be executed when the Charging Station has a permanently attached cable on the Charging Station side.*	**4.** The OCTT responds accordingly.
	Manual Action: *Reconnect the EV and EVSE on EV side.* Note(s): *- If the Charging Station has a permanently attached cable on the Charging Station side, then this step needs to be executed before the configured EVConnectionTimeout expires.*	
	5. The Charging Station sends a **TransactionEventRequest**	**6.** The OCTT responds with a **TransactionEventResponse**
	Note(s): *Optionally the Charging Station can send a StatusNotificationRequest or NotifyEventRequest with status Available*	

Test case name	**Disconnect cable on EV-side - Suspend transaction - Fixed cable connection timeout**
Tool validations	* Step 1: Message: **TransactionEventRequest** - **triggerReason** must be *EVCommunicationLost* - **transactionInfo.chargingState** must be *Idle* - **eventType** must be *Updated* * Step 3: Message: **StatusNotificationRequest** - **connectorStatus** *Available* Message: **NotifyEventRequest** - **eventData[0].trigger** *Delta* - **eventData[0].actualValue** *"Available"* - **eventData[0].component.name** *"Connector"* - **eventData[0].variable.name** *"AvailabilityState"* * Step 5: Message: **TransactionEventRequest** - **triggerReason** must be *EVConnectTimeout* If <Configured TxCtrlr.TxStopPoint> contains *Authorized* then **eventType** must be *Ended* **transactionInfo.stoppedReason** must be *Timeout* else if <Configured TxCtrlr.TxStopPoint> contains *ParkingBayOccupancy* then **eventType** must be *Updated*
	Post scenario validations: N/a

Table 158. Test Case Id: TC_E_28_CS

Test case name	**Check Transaction status - TransactionId unknown**	
Test case Id	TC_E_28_CS	
Use case Id(s)	E14	
Requirement(s)	E14.FR.01	
System under test	Charging Station	
Description	The CSMS is able to request the status of a transaction and to find out whether there are queued transaction-related messages, using the **GetTransactionStatusRequest** message.	
Purpose	To verify if the Charging Station is able to handle receiving a **GetTransactionStatusRequest** for an unknown transactionId.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **GetTransactionStatusResponse**	**1.** The OCTT sends a **GetTransactionStatusRequest** with **transactionId** *<Randomly generated transactionId>*
Tool validations	* Step 2: Message: **GetTransactionStatusResponse** - **ongoingIndicator** must be *false* - **messagesInQueue** must be *false*	
	Post scenario validations: N/a	

Table 159. Test Case Id: TC_E_29_CS

Test case name	**Check Transaction status - Transaction with id ongoing - with message in queue**	
Test case Id	TC_E_29_CS	
Use case Id(s)	E14	
Requirement(s)	E14.FR.02,E14.FR.04	
System under test	Charging Station	
Description	The CSMS is able to request the status of a transaction and to find out whether there are queued transaction-related messages, using the **GetTransactionStatusRequest** message.	
Purpose	To verify if the Charging Station is able to correctly respond to a **GetTransactionStatusRequest** with a transactionId, while there is a message queued belonging to an ongoing transaction with the requested id.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** **SampledDataTxUpdatedMeasurands** is *<Configured transaction_updated_metervalues_measurands>* **SampledDataTxUpdatedInterval** is *<Configured transaction_updated_metervalues_interval>* **OfflineThreshold** is *<Configured RetryBackOffWaitMinimum_duration> + 60.0* **RetryBackOffWaitMinimum** is *<Configured RetryBackOffWaitMinimum_duration>* **RetryBackOffRandomRange** is *0* Note: *<Configured RetryBackOffWaitMinimum_duration> must be greater than <Configured Transaction MeterValues interval>*	
	Memory State: N/a	
	Reusable State(s): **State is** *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
		1. *The OCTT closes the WebSocket connection AND does not accept a reconnect.*
		2. *The OCTT waits a number of seconds equal to <Configured RetryBackOffWaitMinimum_duration>, before accepting a reconnection attempt from the Charging Station.*
	4. The Charging Station responds with a **GetTransactionStatusResponse**	**3.** The OCTT sends a **GetTransactionStatusRequest** with **transactionId** *<Generated transactionId from Before>* Note: *This step will be executed the moment the WebSocket connection is restored.*
	5. The Charging Stations sends a **TransactionEventRequest** Note(s): *- The Charging Station will empty its Transaction message queue. This will contain one or more TransactionEventRequest messages*	**6.** The OCTT responds with a **TransactionEventResponse**
Tool validations	* Step 4: Message: **GetTransactionStatusResponse** - **ongoingIndicator** must be *true* - **messagesInQueue** must be *true* * Step 5: Message: **TransactionEventRequest** - **eventType** must be *Updated* - **meterValues** must be present. - **offline** must be *true*	
	Post scenario validations: N/a	

Table 160. Test Case Id: TC_E_30_CS

Test case name	**Check Transaction status - Transaction with id ongoing - without message in queue**	
Test case Id	TC_E_30_CS	
Use case Id(s)	E14	
Requirement(s)	E14.FR.02,E14.FR.05	
System under test	Charging Station	
Description	The CSMS is able to request the status of a transaction and to find out whether there are queued transaction-related messages, using the **GetTransactionStatusRequest** message.	
Purpose	To verify if the Charging Station is able to correctly respond to a **GetTransactionStatusRequest** with a transactionId, while there is NO message queued belonging to an ongoing transaction with the requested id.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): **State is** *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **GetTransactionStatusResponse**	**1.** The OCTT sends a **GetTransactionStatusRequest** with **transactionId** *<Generated transactionId from Before>*
Tool validations	* Step 2: Message: **GetTransactionStatusResponse** - **ongoingIndicator** must be *true* - **messagesInQueue** must be *false*	
	Post scenario validations: N/a	

Table 161. Test Case Id: TC_E_31_CS

Test case name	**Check Transaction status - Transaction with id ended - with message in queue**	
Test case Id	TC_E_31_CS	
Use case Id(s)	E14	
Requirement(s)	E14.FR.03,E14.FR.04	
System under test	Charging Station	
Description	The CSMS is able to request the status of a transaction and to find out whether there are queued transaction-related messages, using the **GetTransactionStatusRequest** message.	
Purpose	To verify if the Charging Station is able to correctly respond to a **GetTransactionStatusRequest** with a transactionId, while there is a message queued belonging to an ended transaction with the requested id.	
Prerequisite(s)	The following combination of conditions are NOT true: - No local authorization methods are supported AND - TxStopPoint mutability is *false* and only contains Authorized AND - TxCtrlr.StopTxOnEVSideDisconnect mutability is *false* and value is *false* *Note: If conditions 2 and 3 are true, but condition 1 is false, then please configure OCTT configuration <scenario> as local.*	
Before (Preparations)	**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*	
	Memory State: N/a	
	Reusable State(s): **State is** *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
		The OCTT closes the WebSocket connection AND does not accept a reconnect.
	Manual Action: *Present the same idToken as used to start the transaction.*	
	Notes(s): *Only if configured scenario is local*	
	Manual Action: *Disconnect the EV and EVSE.*	
	Manual Action: *Drive EV out of parking bay. (Only needed if TxStopPoint is ParkingBayOccupancy)*	
	Notes(s): *The tool will wait for <Configured Transaction Duration> seconds*	
		The OCTT accepts reconnection attempt from the Charging Station.
	2. The Charging Station responds with a **GetTransactionStatusResponse**	**1.** The OCTT sends a **GetTransactionStatusRequest** with **transactionId** *<Generated transactionId from Before>* Note: *This step will be executed the moment the WebSocket connection is restored.*
	3. The Charging Stations sends a **TransactionEventRequest** Note(s): *- The Charging Station will empty its Transaction message queue. This will contain all TransactionEventRequest messages from the Transaction.*	**4.** The OCTT responds with a **TransactionEventResponse**

Test case name	**Check Transaction status - Transaction with id ended - with message in queue**
Tool validations	* Step 2: Message: **GetTransactionStatusResponse** - **ongoingIndicator** must be *false* - **messagesInQueue** must be *true* * Step 3: Message: **TransactionEventRequest** The tool validations from the reusable states need to be used to verify whether all required TransactionEventRequests have been received. From *StopAuthorized* through *ParkingBayUnoccupied*
	Post scenario validations: N/a

Table 162. Test Case Id: TC_E_32_CS

Test case name	**Check Transaction status - Transaction with id ended - without message in queue**	
Test case Id	TC_E_32_CS	
Use case Id(s)	E14	
Requirement(s)	E14.FR.03,E14.FR.05	
System under test	Charging Station	
Description	The CSMS is able to request the status of a transaction and to find out whether there are queued transaction-related messages, using the **GetTransactionStatusRequest** message.	
Purpose	To verify if the Charging Station is able to correctly respond to a **GetTransactionStatusRequest** with a transactionId, while there is NO message queued belonging to an ended transaction with the requested id.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): **State is** *EnergyTransferStarted*, *ParkingBayUnoccupied*	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **GetTransactionStatusResponse**	**1.** The OCTT sends a **GetTransactionStatusRequest** with **transactionId** *<Generated transactionId from Before>*
Tool validations	* Step 2: Message: **GetTransactionStatusResponse** - **ongoingIndicator** must be *false* - **messagesInQueue** must be *false*	
	Post scenario validations: N/a	

Table 163. Test Case Id: TC_E_33_CS

Test case name	**Check Transaction status - Without transactionId - with message in queue**	
Test case Id	TC_E_33_CS	
Use case Id(s)	E14	
Requirement(s)	E14.FR.06,E14.FR.07	
System under test	Charging Station	
Description	The CSMS is able to request the status of a transaction and to find out whether there are queued transaction-related messages, using the **GetTransactionStatusRequest** message.	
Purpose	To verify if the Charging Station is able to correctly respond to a **GetTransactionStatusRequest** without a transactionId, while there is a message queued.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** **SampledDataTxUpdatedMeasurands** is *<Configured transaction_updated_metervalues_measurands>* **SampledDataTxUpdatedInterval** is *<Configured transaction_updated_metervalues_interval>* **OfflineThreshold** is *<Configured RetryBackOffWaitMinimum_duration> + 60.0* **RetryBackOffWaitMinimum** is *<Configured RetryBackOffWaitMinimum_duration>* **RetryBackOffRandomRange** is *0* Note: *<Configured RetryBackOffWaitMinimum_duration> must be greater than <Configured Transaction MeterValues interval>*	
	Memory State: N/a	
	Reusable State(s): **State is** *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
		1. *The OCTT closes the WebSocket connection AND does not accept a reconnect.*
		2. *The OCTT waits a number of seconds equal to <Configured RetryBackOffWaitMinimum_duration>, before accepting a reconnection attempt from the Charging Station.*
	4. The Charging Station responds with a **GetTransactionStatusResponse**	**3.** The OCTT sends a **GetTransactionStatusRequest** with **transactionId** omitted Note: *This step will be executed the moment the WebSocket connection is restored.*
	5. The Charging Stations sends a **TransactionEventRequest** Note(s): *- The Charging Station will empty its Transaction message queue. This will contain one or more TransactionEventRequest messages*	**6.** The OCTT responds with a **TransactionEventResponse**
Tool validations	* Step 4: Message: **GetTransactionStatusResponse** - **ongoingIndicator** must be omitted - **messagesInQueue** must be *true* * Step 5: Message: **TransactionEventRequest** - **eventType** must be *Updated* - **meterValues** must be present. - **offline** must be *true*	
	Post scenario validations: N/a	

Table 164. Test Case Id: TC_E_34_CS

Test case name	**Check Transaction status - Without transactionId - without message in queue**	
Test case Id	TC_E_34_CS	
Use case Id(s)	E14	
Requirement(s)	E14.FR.06,E14.FR.08	
System under test	Charging Station	
Description	The CSMS is able to request the status of a transaction and to find out whether there are queued transaction-related messages, using the **GetTransactionStatusRequest** message.	
Purpose	To verify if the Charging Station is able to correctly respond to a **GetTransactionStatusRequest** without a transactionId, while there is NO message queued.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **GetTransactionStatusResponse**	**1.** The OCTT sends a **GetTransactionStatusRequest** with **transactionId** omitted
Tool validations	* Step 2: Message: **GetTransactionStatusResponse** - **ongoingIndicator** must be omitted - **messagesInQueue** must be *false*	
	Post scenario validations: N/a	

Table 165. Test Case Id: TC_E_40_CS

Test case name	**Offline Behaviour - Connection loss during transaction**	
Test case Id	TC_E_40_CS	
Use case Id(s)	E11	
Requirement(s)	E11.FR.01,E11.FR.02,E11.FR.06	
System under test	Charging Station	
Description	The Charging Station queues TransactionEvent messages to inform the CSMS that a transaction occurred while the Charging Station was Offline.	
Purpose	To verify if the Charging Station is able to queue TransactionEvent messages while it is offline.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** **SampledDataTxUpdatedMeasurands** is *<Configured transaction_updated_metervalues_measurands>* **SampledDataTxUpdatedInterval** is *<Configured transaction_updated_metervalues_interval>* **SampledDataEnabled** is *true* **OfflineThreshold** is *<Configured RetryBackOffWaitMinimum_duration> + 60.0* **RetryBackOffWaitMinimum** is *<Configured RetryBackOffWaitMinimum_duration>* **RetryBackOffRandomRange** is *0* Note: *<Configured RetryBackOffWaitMinimum_duration> must be greater than <Configured Transaction MeterValues interval>*	
	Memory State: N/a	
	Reusable State(s): **State is** *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
		1. *The OCTT closes the WebSocket connection AND does not accept a reconnect.*
		2. *The OCTT waits a number of seconds equal to <Configured RetryBackOffWaitMinimum_duration>, before accepting a reconnection attempt from the Charging Station.*
	3. The Charging Stations sends a **TransactionEventRequest** Note(s): - *The Charging Station will empty its Transaction message queue. This will contain one or more TransactionEventRequest messages*	**4.** The OCTT responds with a **TransactionEventResponse**
Tool validations	* Step 3: All messages: **TransactionEventRequest** - **eventType** must be *Updated* - **meterValues** must be present. - **offline** must be *true*	
	Post scenario validations: N/a	

Table 166. Test Case Id: TC_E_41_CS

Test case name	**Retry sending transaction message when failed - Max retry count reached**	
Test case Id	TC_E_41_CS	
Use case Id(s)	E13	
Requirement(s)	E13.FR.01,E13.FR.02,E13.FR.03,E13.FR.04	
System under test	Charging Station	
Description	There are situations/issues why a CSMS might not accept a transaction related message, or does not reply within the MessageTimeout. Most are error scenarios. When something like this happens, the Charging Station SHALL retry the messages a couple of times.	
Purpose	To verify if the Charging Station is able to resend TransactionEvent messages until the max retry count is reached, when the CSMS does not reply.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** **MessageAttemptsTransactionEvent** is *<Configured message_attempts_transaction_event>* (Must be > 1) **MessageAttemptIntervalTransactionEvent** is *<Configured message_attempts_transaction_event_interval>*	
	Memory State: N/a	
	Reusable State(s): **State is** *Authorized* **State is** *EVConnectedPreSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
	Note(s): *Step 1, 2, 3, & 4 are optional*	
	1. The Charging Stations sends a **TransactionEventRequest** with: - **triggerReason** *SignedDataReceived*	**2.** The OCTT responds with a **TransactionEventResponse**
	3. The Charging Stations sends a **TransactionEventRequest** with: - **triggerReason** *ChargingStateChanged* - **chargingState** *SuspendedEVSE*	**4.** The OCTT responds with a **TransactionEventResponse**
	Note(s): *Step 5 is repeated for the configured number of times*	
	5. The Charging Stations sends a **TransactionEventRequest**	
Tool validations	* Step 1: - **triggerReason** *SignedDataReceived* * Step 3: - **triggerReason** *ChargingStateChanged* - **chargingState** *SuspendedEVSE* * Step 5: - Needs to be sent 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).	
	Post scenario validations: N/a	

Table 167. Test Case Id: TC_E_50_CS

Test case name	**Retry sending transaction message when failed - Max retry count reached - CallError**	
Test case Id	TC_E_50_CS	
Use case Id(s)	E13	
Requirement(s)	E13.FR.01,E13.FR.02,E13.FR.03,E13.FR.04	
System under test	Charging Station	
Description	There are situations/issues why a CSMS might not accept a transaction related message, or does not reply within the MessageTimeout. Most are error scenarios. When something like this happens, the Charging Station SHALL retry the messages a couple of times.	
Purpose	To verify if the Charging Station is able to resend TransactionEvent messages until the max retry count is reached, when the CSMS does not reply.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** **MessageAttemptsTransactionEvent** is *<Configured message_attempts_transaction_event>* (Must be > 1) **MessageAttemptIntervalTransactionEvent** is *<Configured message_attempts_transaction_event_interval>*	
	Memory State: N/a	
	Reusable State(s): **State is** *Authorized* **State is** *EVConnectedPreSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
	Note(s): *Step 1, 2, 3, & 4 are optional*	
	1. The Charging Stations sends a **TransactionEventRequest** with: - **triggerReason** *SignedDataReceived*	**2.** The OCTT responds with a **TransactionEventResponse**
	3. The Charging Stations sends a **TransactionEventRequest** with: - **triggerReason** *ChargingStateChanged* - **chargingState** *SuspendedEVSE*	**4.** The OCTT responds with a **TransactionEventResponse**
	Note(s): *Step 5 is repeated for the configured number of times*	
	5. The Charging Stations sends a **TransactionEventRequest**	**6.** The OCTT responds with a **CallError** with **errorCode** *InternalError*
Tool validations	* Step 1: - **triggerReason** *SignedDataReceived* * Step 3: - **triggerReason** *ChargingStateChanged* - **chargingState** *SuspendedEVSE* * Step 5: - Needs to be sent a number of times equal to *<Configured message_attempts_transaction_event>* with an interval of the *<Configured message_attempts_transaction_event_interval>* multiplied by the number of preceding transmissions of this same message. - The OCTT waits an additional MessageAttemptsTransactionEvent iteration where the interval is multiplied again, to validate if the Charging Station stops resending the TransactionRequest message(s).	
	Post scenario validations: N/a	

Table 168. Test Case Id: TC_E_42_CS

Test case name	**Retry sending transaction message when failed - Success before reaching the max retry count**	
Test case Id	TC_E_42_CS	
Use case Id(s)	E13	
Requirement(s)	E13.FR.01,E13.FR.02,E13.FR.03	
System under test	Charging Station	
Description	There are situations/issues why a CSMS might not accept a transaction related message, or does not reply within the MessageTimeout. Most are error scenarios. When something like this happens, the Charging Station SHALL retry the messages a couple of times.	
Purpose	To verify if the Charging Station is able to resend TransactionEvent messages when the CSMS does not reply and stops resending after getting a response before the max retry count is reached.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** **MessageAttemptsTransactionEvent** is *<Configured message_attempts_transaction_event>* (Must be > 2) **MessageAttemptIntervalTransactionEvent** is *<Configured message_attempts_transaction_event_interval>*	
	Memory State: N/a	
	Reusable State(s): **State is** *Authorized* **State is** *EVConnectedPreSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
	Note(s): *Step 1, 2, 3, & 4 are optional*	
	1. The Charging Stations sends a **TransactionEventRequest** with: - **triggerReason** *SignedDataReceived*	**2.** The OCTT responds with a **TransactionEventResponse**
	3. The Charging Stations sends a **TransactionEventRequest** with: - **triggerReason** *ChargingStateChanged* - **chargingState** *SuspendedEVSE*	**4.** The OCTT responds with a **TransactionEventResponse**
	Note(s): *The tool will ignore the first request and only respond to the second request*	
	5. The Charging Stations sends a **TransactionEventRequest**	**6.** The OCTT responds with a **TransactionEventResponse**
Tool validations	* Step 1: - **triggerReason** *SignedDataReceived* * Step 3: - **triggerReason** *ChargingStateChanged* - **chargingState** *SuspendedEVSE* * 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).	
	Post scenario validations: N/a	

Table 169. Test Case Id: TC_E_51_CS

Test case name	**Retry sending transaction message when failed - Success before reaching the max retry count - CallError**	
Test case Id	TC_E_51_CS	
Use case Id(s)	E13	
Requirement(s)	E13.FR.01,E13.FR.02,E13.FR.03	
System under test	Charging Station	
Description	There are situations/issues why a CSMS might not accept a transaction related message, or does not reply within the MessageTimeout. Most are error scenarios. When something like this happens, the Charging Station SHALL retry the messages a couple of times.	
Purpose	To verify if the Charging Station is able to resend TransactionEvent messages when the CSMS does not reply and stops resending after getting a response before the max retry count is reached.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** **MessageAttemptsTransactionEvent** is *<Configured message_attempts_transaction_event>* (Must be > 2) **MessageAttemptIntervalTransactionEvent** is *<Configured message_attempts_transaction_event_interval>*	
	Memory State: N/a	
	Reusable State(s): **State is** *Authorized* **State is** *EVConnectedPreSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
	Note(s): *Step 1, 2, 3, & 4 are optional*	
	1. The Charging Stations sends a **TransactionEventRequest** with: - **triggerReason** *SignedDataReceived*	**2.** The OCTT responds with a **TransactionEventResponse**
	3. The Charging Stations sends a **TransactionEventRequest** with: - **triggerReason** *ChargingStateChanged* - **chargingState** *SuspendedEVSE*	**4.** The OCTT responds with a **TransactionEventResponse**
	Note(s): *The tool will send a CallError with errorCode InternalError to all requests except for the second request, there a TransactionEventResponse is send*	
	5. The Charging Stations sends a **TransactionEventRequest**	**6.** The OCTT responds with a **TransactionEventResponse**
Tool validations	* Step 1: - **triggerReason** *SignedDataReceived* * Step 3: - **triggerReason** *ChargingStateChanged* - **chargingState** *SuspendedEVSE* * 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).	
	Post scenario validations: N/a	

Table 170. Test Case Id: TC_E_43_CS

Test case name	**Offline Behaviour - Transaction during offline period**	
Test case Id	TC_E_43_CS	
Use case Id(s)	E12	
Requirement(s)	E12.FR.01,E12.FR.02,E12.FR.06	
System under test	Charging Station	
Description	The Charging Station queues TransactionEvent messages to inform the CSMS that a transaction occurred while the Charging Station was Offline.	
Purpose	To verify if the Charging Station is able to queue TransactionEvent messages while it was offline.	
Prerequisite(s)	The Charging Station supports authorization methods other than NoAuthorization	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *TransactionEventsInQueueEnded*	
	2. The Charging Stations sends a **TransactionEventRequest** Note(s): *- The Charging Station will empty its Transaction message queue. This will contain one or more TransactionEventRequest messages*	**3.** The OCTT responds with a **TransactionEventResponse**
Tool validations	* Step 2: All messages: **TransactionEventRequest** - **offline** must be *true* One of the messages: **TransactionEventRequest** - **eventType** *Started* One of the messages: **TransactionEventRequest** - **eventType** *Ended*	
	Post scenario validations: N/a	

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

Table 171. Test Case Id: TC_E_44_CS

Test case name	**Offline Behaviour - Stop transaction during offline period**	
Test case Id	TC_E_44_CS	
Use case Id(s)	E08	
Requirement(s)	E08.FR.01,E08.FR.04,E08.FR.05,E08.FR.06,E08.FR.07,E08.FR.08	
System under test	Charging Station	
Description	The Charging Station queues TransactionEvent messages to inform the CSMS that a transaction occurred while the Charging Station was Offline.	
Purpose	To verify if the Charging Station is able to queue TransactionEvent messages when the transaction stopped while the Charging Station was offline.	
Prerequisite(s)	N/a	
Before (Preparations)	**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>*	
	Memory State: N/a	
	Reusable State(s): **State is** *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
		1. *The OCTT closes the WebSocket connection AND does not accept a reconnect.*
	Manual Action: *Present the same idToken as used to start the transaction.*	
	Manual Action: *Disconnect the EV and EVSE.*	
	Manual Action: *Drive EV out of parking bay. (Only needed if TxStopPoint is ParkingBayOccupancy)*	
		2. *The OCTT accepts the reconnection attempt from the Charging Station.*
	3. The Charging Stations sends a **TransactionEventRequest** Note(s): *- The Charging Station will empty its Transaction message queue. This will contain one or more TransactionEventRequest messages*	**4.** The OCTT responds with a **TransactionEventResponse**
Tool validations	* Step 3: All messages: **TransactionEventRequest** - **offline** must be *true* One of the messages: **TransactionEventRequest** - **eventType** *Ended*	
	Post scenario validations: N/a	

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

Table 172. Test Case Id: TC_E_45_CS

Test case name	**Offline Behaviour - Stop transaction during offline period - Same GroupId**	
Test case Id	TC_E_45_CS	
Use case Id(s)	E08	
Requirement(s)	E08.FR.02,E08.FR.04,E08.FR.05,E08.FR.06,E08.FR.07,E08.FR.08	
System under test	Charging Station	
Description	The Charging Station queues TransactionEvent messages to inform the CSMS that a transaction occurred while the Charging Station was Offline.	
Purpose	To verify if the Charging Station is able to queue TransactionEvent messages when the transaction stopped by an idToken with the same groupIdToken, while the Charging Station was offline.	
Prerequisite(s)	- The Charging Station supports Authorization cache OR Local Authorization List. - The Charging Station supports authorization methods other than NoAuthorization	
Before (Preparations)	**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>*	
	Memory State: *IdTokenCached* for *<Configured valid idtoken fields2>* with *<Configured GroupIdToken>* *IdTokenLocalAuthList* for *<Configured valid idtoken fields2>* with *<Configured GroupIdToken>*	
	Reusable State(s): **State is** *Authorized* with *<Configured GroupIdToken>* Then proceed to state *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
		1. *The OCTT closes the WebSocket connection AND does not accept a reconnect.*
	Manual Action: *Present <Configured valid idtoken fields2>.*	
	Manual Action: *Disconnect the EV and EVSE.*	
	Manual Action: *Drive EV out of parking bay. (Only needed if TxStopPoint is ParkingBayOccupancy)*	
		2. *The OCTT accepts the reconnection attempt from the Charging Station.*
	3. The Charging Stations sends a **TransactionEventRequest** Note(s): *- The Charging Station will empty its Transaction message queue. This will contain one or more TransactionEventRequest messages*	**4.** The OCTT responds with a **TransactionEventResponse**
Tool validations	* Step 3: All messages: **TransactionEventRequest** - **offline** must be *true* One of the messages: **TransactionEventRequest** - **eventType** *Ended*	
	Post scenario validations: N/a	

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

Table 173. Test Case Id: TC_E_46_CS

Test case name	**End of charging process 15118**	
Test case Id	TC_E_46_CS	
Use case Id(s)	E15	
Requirement(s)	E15.FR.04, E15.FR.05	
System under test	Charging Station	
Description	After receiving a SessionStopReq(Terminate) message from the EV, the Charging Station informs the CSMS that the authorization of the charging session has been stopped (by the EV). Depending on TxStopPoint this will also end the transaction.	
Purpose	To verify whether the Charging Station is able to inform the CSMS that authorization of the charging session has been stopped (by the EV) and depending on TxStopPoint end the transaction.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): **State is** *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	Note: The Charging Station receives a SessionStopReq(Terminate) message from the EV.	
	1. The Charging Station sends a **TransactionEventRequest**	**2.** The OCTT responds with a **TransactionEventResponse**
Tool validations	* Step 1: Message: **TransactionEventRequest** If <Configured TxStopPoint> contains "Authorized" or "PowerPathClosed" or "EnergyTransfer": - **eventType** is *Ended* - **triggerReason** is *StopAuthorized* - **transactionInfo.stoppedReason** is *StoppedByEV* - **transactionInfo.chargingState** is *EVConnected* If <Configured TxStopPoint> does not contain "Authorized" or "PowerPathClosed" or "EnergyTransfer": - **eventType** is *Updated* - **triggerReason** = *StopAuthorized* - **transactionInfo.chargingState** is *EVConnected*	
	Post scenario validations: N/a	

2.7. F Remote Control

Table 174. Test Case Id: TC_F_01_CS

Test case name	**Remote start transaction - Cable plugin first**	
Test case Id	TC_F_01_CS	
Use case Id(s)	F01	
Requirement(s)	F01.FR.03, F01.FR.04, F01.FR.05, F01.FR.13, F01.FR.17, F01.FR.19, F02.FR.01	
System under test	Charging Station	
Description	OCPP 2.x.x allows an EV driver to either first connect the EV and EVSE OR wait for/trigger a RequestStartTransactionRequest. Both sequences will result in being able to charge.	
Purpose	To verify if the Charging Station is able to start a charging session when the EV driver first connects the EV and EVSE, before receiving a RequestStartTransactionRequest message.	
Prerequisite(s)	- The Charging Station does NOT have a cable lock, which prevents the EV driver to connect the EV and EVSE before authorization.	
Before (Preparations)	**Configuration State:** **AuthCtrlr.AuthEnabled** is *true* (If implemented AND ReadWrite) **AuthCtrlr.DisableRemoteAuthorization** is *false* (If implemented)	
	Memory State: N/a	
	Reusable State(s): **State is** *EVConnectedPreSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *Authorized* (remote)	
	2. Execute **Reusable State** *EnergyTransferStarted*	
Tool validations	N/a	
	Post scenario validations: N/a	

Table 175. Test Case Id: TC_F_02_CS

Test case name	**Remote start transaction - Remote start first - AuthorizeRemoteStart is true**	
Test case Id	TC_F_02_CS	
Use case Id(s)	F02	
Requirement(s)	F02.FR.01, F01.FR.01	
System under test	Charging Station	
Description	OCPP 2.x.x allows an EV driver to either first wait for/trigger a RequestStartTransactionRequest OR connect the EV and EVSE. Both sequences will result in being able to charge.	
Purpose	To verify if the Charging Station is able to start a charging session when the Charging Stations receives a RequestStartTransactionRequest message (while AuthorizeRemoteStart is true), before the EV driver connects the EV and EVSE (within the connectionTimeout). The Charging Station has to authorize beforehand like a local action to start a transaction.	
Prerequisite(s)	- **AuthEnabled** is NOT implemented with mutability ReadOnly and the value set to false AND - **AuthorizeRemoteStart** is NOT implemented with mutability ReadOnly and the value set to false	
Before (Preparations)	**Configuration State:** **AuthCtrlr.AuthEnabled** is *true* (If implemented AND ReadWrite) **AuthCtrlr.DisableRemoteAuthorization** is *false* (If implemented) **AuthorizeRemoteStart** is *true* (If ReadWrite)	
	Memory State: N/a	
	Reusable State(s): **State is** *ParkingBayOccupied* (Optional state)	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *Authorized* (remote)	
	2. Execute **Reusable State** *EnergyTransferStarted*	
Tool validations	N/a	
	Post scenario validations: N/a	

Table 176. Test Case Id: TC_F_03_CS

Test case name	**Remote start transaction - Remote start first - AuthorizeRemoteStart is false**	
Test case Id	TC_F_03_CS	
Use case Id(s)	F02	
Requirement(s)	F02.FR.01, F01.FR.02	
System under test	Charging Station	
Description	OCPP 2.x.x allows an EV driver to either first wait for/trigger a RequestStartTransactionRequest OR connect the EV and EVSE. Both sequences will result in being able to charge.	
Purpose	To verify if the Charging Station is able to start a charging session when the Charging Stations receives a RequestStartTransactionRequest message (while AuthorizeRemoteStart is false), before the EV driver connects the EV and EVSE (within the connectionTimeout). The Charging station does NOT have to authorize beforehand like a local action to start a transaction.	
Prerequisite(s)	**AuthorizeRemoteStart** is NOT implemented with mutability ReadOnly and the value set to true	
Before (Preparations)	**Configuration State:** **AuthCtrlr.AuthEnabled** is *true* (If implemented AND ReadWrite) **AuthCtrlr.DisableRemoteAuthorization** is *false* (If implemented) **AuthorizeRemoteStart** is *false* (If ReadWrite)	
	Memory State: N/a	
	Reusable State(s): **State is** *ParkingBayOccupied* (Optional state)	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *Authorized* (remote)	
	2. Execute **Reusable State** *EnergyTransferStarted*	
Tool validations	N/a	
	Post scenario validations: N/a	

Table 177. Test Case Id: TC_F_04_CS

Test case name	**Remote start transaction - Remote start first - Cable plugin timeout**	
Test case Id	TC_F_04_CS	
Use case Id(s)	F02, E03	
Requirement(s)	F02.FR.01, E03.FR.01, E03.FR.05	
System under test	Charging Station	
Description	OCPP 2.x.x allows an EV driver to either first wait for/trigger a RequestStartTransactionRequest OR connect the EV and EVSE. Both sequences will result in being able to charge.	
Purpose	To verify if the Charging Station is able to deauthorize the transaction after the **EVConnectionTimeout** has been reached.	
Prerequisite(s)	The Charging Station supports **TxCtrlr.TxStartPoint** *ParkingBayOccupancy* OR *Authorized*.	
Before (Preparations)	**Configuration State:** - **TxCtrlr.EVConnectionTimeOut** is *<Configured ev_connection_timeout>* - **AuthCtrlr.AuthEnabled** is *true* (If implemented AND ReadWrite) **AuthCtrlr.DisableRemoteAuthorization** is *false* (If implemented) - **TxCtrlr.TxStartPoint** is *ParkingBayOccupancy* OR *Authorized*	
	Memory State: N/a	
	Reusable State(s): **State is** *Authorized* (remote)	
Main (Test scenario)	**Charging Station**	**CSMS**
	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*	
	3. Execute **Reusable State** *Authorized* (remote) Note(s): *- This step is executed to verify if the EVSE is actually ready to start another charging session.*	
Tool validations	* Step 1: Message: **TransactionEventRequest** - **triggerReason** must be *EVConnectTimeout* - **eventType** must be *Ended*	
	Post scenario validations: N/a	

Table 178. Test Case Id: TC_F_05_CS

Test case name	**Remote unlock Connector - With ongoing transaction**	
Test case Id	TC_F_05_CS	
Use case Id(s)	F05	
Requirement(s)	F05.FR.01, F05.FR.02	
System under test	Charging Station	
Description	This test case describes how the CSMS can be requested to sent an UnlockConnectorRequest to the charging station. It sometimes happens that a connector of a Charging Station socket does not unlock correctly. This happens most of the time when there is tension on the charging cable. This means the driver cannot unplug his charging cable from the Charging Station. To help a driver, the CSO can send a UnlockConnectorRequest to the Charging Station. The Charging Station will then try to unlock the connector again.	
Purpose	To verify if the Chargin Station is able to ignore the UnlockConnectorRequest whith an ongoing transaction as described at the OCPP specification.	
Prerequisite(s)	The Charging Station has a connector lock.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: Transaction is ongoing on <Configured Connector> State is EnergyTransferStarted	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **UnlockConnectorResponse**	**1.** The OCTT sends a **UnlockConnectorRequest** with **evseId** *<Configured evseId>* **connectorId** *<Configured connectorId>*
Tool validations	* Step 2: Message **UnlockConnectorResponse** - **status** *OngoingAuthorizedTransaction*	
	Post scenario validations: - N/a	

Table 179. Test Case Id: TC_F_06_CS

Test case name	**Remote unlock Connector - Without ongoing transaction - Accepted**	
Test case Id	TC_F_06_CS	
Use case Id(s)	F05	
Requirement(s)	F05.FR.01, F05.FR.04	
System under test	Charging Station	
Description	This test case describes how the CSMS can be requested to sent an UnlockConnectorRequest to the charging station. It sometimes happens that a connector of a Charging Station socket does not unlock correctly. This happens most of the time when there is tension on the charging cable. This means the driver cannot unplug his charging cable from the Charging Station. To help a driver, the CSO can send a UnlockConnectorRequest to the Charging Station. The Charging Station will then try to unlock the connector again.	
Purpose	To verify if the Charging Station is able to successfully unlock a connector without ongoing transaction as described in the OCPP specification.	
Prerequisite(s)	The Charging Station has a connector lock.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **UnlockConnectorResponse**	**1.** The OCTT sends a **UnlockConnectorRequest** with **evseId** *<Configured evseId>* **connectorId** *<Configured connectorId>*
Tool validations	* Step 2: Message **UnlockConnectorResponse** - **status** *Unlocked*	
	Post scenario validations: - N/a	

Table 180. Test Case Id: TC_F_07_CS

Test case name	**Remote unlock Connector - Without ongoing transaction - No cable connected**	
Test case Id	TC_F_07_CS	
Use case Id(s)	F05	
Requirement(s)	F05.FR.01, F05.FR.06	
System under test	Charging Station	
Description	This test case describes how the CSMS can be requested to sent an UnlockConnectorRequest to the charging station. It sometimes happens that a connector of a Charging Station socket does not unlock correctly. This happens most of the time when there is tension on the charging cable. This means the driver cannot unplug his charging cable from the Charging Station. To help a driver, the CSO can send a UnlockConnectorRequest to the Charging Station. The Charging Station will then try to unlock the connector again.	
Purpose	To verify if the Chargin Station is able to perform the remote unlock connector mechanism and report the result without ongoing transaction while no cable is connected as described at the OCPP specification.	
Prerequisite(s)	The Charging Station has a connector lock.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: No cable connected at <Configured Connector>	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **UnlockConnectorResponse**	**1.** The OCTT sends a **UnlockConnectorRequest** with **evseId** *<Configured evseId>* **connectorId** *<Configured connectorId>*
Tool validations	* Step 2: Message **UnlockConnectorResponse** - **status** *Unlocked*	
	Post scenario validations: - N/a	

Table 181. Test Case Id: TC_F_08_CS

Test case name	**Remote stop transaction - Success**	
Test case Id	TC_F_08_CS	
Use case Id(s)	F03	
Requirement(s)	F03.FR.02, F03.FR.03, F03.FR.07, F03.FR.09	
System under test	Charging Station	
Description	The CSMS is able to stop a charging session remotely by sending a RequestStopTransactionRequest to the Charging Station.	
Purpose	To verify if the Charging Station is able to stop a charging session when it receives a RequestStopTransactionRequest message.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): **State is** *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *StopAuthorized* (remote)	
Tool validations	N/a	
	Post scenario validations: N/a	

Table 182. Test Case Id: TC_F_09_CS

Test case name	**Remote stop transaction - Rejected**	
Test case Id	TC_F_09_CS	
Use case Id(s)	F03	
Requirement(s)	F03.FR.08	
System under test	Charging Station	
Description	The CSMS is able to stop a charging session remotely by sending a RequestStopTransactionRequest to the Charging Station.	
Purpose	To verify if the Charging Station will reject a RequestStopTransactionRequest message, if it contains a transactionId that cannot be matched to an active transaction.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): **State is** *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **RequestStopTransactionResponse**	**1.** The OCTT sends a **RequestStopTransactionRequest** with **transactionId** *<Different transactionId than provided by the Charging Station in **TransactionEventRequest**>*
Tool validations	* Step 2: Message: **RequestStopTransactionResponse** - **status** must be *Rejected*	
	Post scenario validations: N/a	

Table 183. Test Case Id: TC_F_10_CS

Test case name	**Remote unlock Connector - Without ongoing transaction - UnknownConnector**	
Test case Id	TC_F_10_CS	
Use case Id(s)	F05	
Requirement(s)	F05.FR.03	
System under test	Charging Station	
Description	This test case describes how the CSMS can be requested to sent an UnlockConnectorRequest to the charging station. It sometimes happens that a connector of a Charging Station socket does not unlock correctly. This happens most of the time when there is tension on the charging cable. This means the driver cannot unplug his charging cable from the Charging Station. To help a driver, the CSO can send a UnlockConnectorRequest to the Charging Station. The Charging Station will then try to unlock the connector again.	
Purpose	To verify if the Charging Station is able to respond with a UnlockConnectorRequest with status *UnknownConnector* when the requested connector is unknown as described in the OCPP specification.	
Prerequisite(s)	The Charging Station has a connector lock.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **UnlockConnectorResponse**	**1.** The OCTT sends a **UnlockConnectorRequest** with **evseId** *<Configured evseId>* **connectorId** *999*
Tool validations	* Step 2: Message **UnlockConnectorResponse** - **status** *UnknownConnector*	
	Post scenario validations: - N/a	

Table 184. Test Case Id: TC_F_11_CS

Test case name	**Trigger message - MeterValues - Specific EVSE**	
Test case Id	TC_F_11_CS	
Use case Id(s)	F06	
Requirement(s)	F06.FR.02,F06.FR.04,F06.FR.05,F06.FR.06,F06.FR.10	
System under test	Charging Station	
Description	The CSMS can request a Charging Station to send Charging Station-initiated messages. In the request the CSMS indicates which message it wishes to receive.	
Purpose	To verify if the Charging Station is able to send a MeterValuesRequest message for a specific EVSE, after receiving a TriggerMessageRequest message.	
Prerequisite(s)	The Charging Station supports sending MeterValues triggered by a TriggerMessageRequest.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **TriggerMessageResponse**	**1.** The OCTT sends a **TriggerMessageRequest** With requestedMessage *MeterValues* evse.id *<Configured evseId>*
	3. The Charging Station sends a **MeterValuesRequest**	**4.** The OCTT responds with a **MeterValuesResponse**
Tool validations	* Step 2: Message: **TriggerMessageResponse** - **status** must be *Accepted* * Step 3: Message: **MeterValuesRequest** - **evseId** must be *<Configured evseId>* - **meterValue[0].sampledValue[0].context** must be *Trigger*	
	Post scenario validations: N/a	

Table 185. Test Case Id: TC_F_12_CS

Test case name	**Trigger message - MeterValues - All EVSE**	
Test case Id	TC_F_12_CS	
Use case Id(s)	F06	
Requirement(s)	F06.FR.04,F06.FR.05,F06.FR.06,F06.FR.10,F06.FR.11	
System under test	Charging Station	
Description	The CSMS can request a Charging Station to send Charging Station-initiated messages. In the request the CSMS indicates which message it wishes to receive.	
Purpose	To verify if the Charging Station is able to send a MeterValuesRequest message for all EVSE, after receiving a TriggerMessageRequest message.	
Prerequisite(s)	The Charging Station supports sending MeterValues triggered by a TriggerMessageRequest.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **TriggerMessageResponse**	**1.** The OCTT sends a **TriggerMessageRequest** With requestedMessage *MeterValues* evse is omitted
	3. The Charging Station sends a **MeterValuesRequest** Note(s): *- This step needs to be executed for every EVSE.*	**4.** The OCTT responds with a **MeterValuesResponse**
Tool validations	* Step 2: Message: **TriggerMessageResponse** - **status** must be *Accepted* * Step 3: Message: **MeterValuesRequest** - **meterValue[0].sampledValue[0].context** must be *Trigger*	
	Post scenario validations: N/a	

Table 186. Test Case Id: TC_F_13_CS

Test case name	**Trigger message - TransactionEvent - Specific EVSE**	
Test case Id	TC_F_13_CS	
Use case Id(s)	F06	
Requirement(s)	F06.FR.02,F06.FR.04,F06.FR.05,F06.FR.07,F06.FR.10	
System under test	Charging Station	
Description	The CSMS can request a Charging Station to send Charging Station-initiated messages. In the request the CSMS indicates which message it wishes to receive.	
Purpose	To verify if the Charging Station is able to send a TransactionEventRequest message for a specific EVSE, after receiving a TriggerMessageRequest message.	
Prerequisite(s)	The Charging Station supports sending TransactionEvents triggered by a TriggerMessageRequest.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): **State is** *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **TriggerMessageResponse**	**1.** The OCTT sends a **TriggerMessageRequest** With requestedMessage *TransactionEvent* evse.id *<Configured evseId>*
	3. The Charging Station sends a **TransactionEventRequest**	**4.** The OCTT responds with a **TransactionEventResponse**
Tool validations	* Step 2: Message: **TriggerMessageResponse** - **status** must be *Accepted* * Step 3: Message: **TransactionEventRequest** - **evse.id** must be *omitted* or *<Configured evseId>* - **triggerReason** must be *Trigger* - **transactionInfo.chargingState** must be *Charging* - **meterValue** must be present - **meterValue[0].sampledValue[0].context** must be *Trigger*	
	Post scenario validations: N/a	

Table 187. Test Case Id: TC_F_14_CS

Test case name	**Trigger message - TransactionEvent - All EVSE**	
Test case Id	TC_F_14_CS	
Use case Id(s)	F06	
Requirement(s)	F06.FR.04,F06.FR.05,F06.FR.07,F06.FR.10,F06.FR.11	
System under test	Charging Station	
Description	The CSMS can request a Charging Station to send Charging Station-initiated messages. In the request the CSMS indicates which message it wishes to receive.	
Purpose	To verify if the Charging Station is able to send a TransactionEventRequest message for all EVSE, after receiving a TriggerMessageRequest message.	
Prerequisite(s)	The Charging Station supports sending TransactionEvents triggered by a TriggerMessageRequest.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): **State is** *EnergyTransferStarted* for all EVSE	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **TriggerMessageResponse**	**1.** The OCTT sends a **TriggerMessageRequest** With requestedMessage *TransactionEvent* evse is omitted
	3. The Charging Station sends a **TransactionEventRequest** Note(s): - *This step needs to be executed for every EVSE.*	**4.** The OCTT responds with a **TransactionEventResponse**
Tool validations	* Step 2: Message: **TriggerMessageResponse** - **status** must be *Accepted* * Step 3: Message: **TransactionEventRequest** - **evse.id** must be *<Configured evseId>* - **triggerReason** must be *Trigger* - **transactionInfo.chargingState** must be *Charging* - **meterValue** must be present - **meterValue[0].sampledValue[0].context** must be *Trigger*	
	Post scenario validations: N/a	

Table 188. Test Case Id: TC_F_15_CS

Test case name	**Trigger message - LogStatusNotification - Idle**	
Test case Id	TC_F_15_CS	
Use case Id(s)	F06	
Requirement(s)	F06.FR.04,F06.FR.05,F06.FR.10,F06.FR.15	
System under test	Charging Station	
Description	The CSMS can request a Charging Station to send Charging Station-initiated messages. In the request the CSMS indicates which message it wishes to receive.	
Purpose	To verify if the Charging Station is able to send a LogStatusNotificationRequest with status Idle, after receiving a TriggerMessageRequest while NOT uploading a log file.	
Prerequisite(s)	The Charging Station supports sending LogStatusNotifications triggered by a TriggerMessageRequest.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **TriggerMessageResponse**	**1.** The OCTT sends a **TriggerMessageRequest** With requestedMessage *LogStatusNotification*
	3. The Charging Station sends a **LogStatusNotificationRequest**	**4.** The OCTT responds with a **LogStatusNotificationResponse**
Tool validations	* Step 2: Message: **TriggerMessageResponse** - **status** must be *Accepted* * Step 3: Message: **LogStatusNotificationRequest** - **status** must be *Idle*	
	Post scenario validations: N/a	

Table 189. Test Case Id: TC_F_16_CS

Test case name	**Trigger message - LogStatusNotification - Uploading**	
Test case Id	TC_F_16_CS	
Use case Id(s)	F06	
Requirement(s)	F06.FR.04,F06.FR.05,F06.FR.10,F06.FR.14	
System under test	Charging Station	
Description	The CSMS can request a Charging Station to send Charging Station-initiated messages. In the request the CSMS indicates which message it wishes to receive.	
Purpose	To verify if the Charging Station is able to send a LogStatusNotificationRequest with status Uploading, after receiving a TriggerMessageRequest while uploading a log file.	
Prerequisite(s)	The Charging Station supports sending LogStatusNotifications triggered by a TriggerMessageRequest.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **GetLogResponse**	**1.** The OCTT sends a **GetLogRequest** With **logType** *DiagnosticsLog* **log.remoteLocation** is *<Configured log_location>*
	3. The Charging Station sends a **LogStatusNotificationRequest**	**4.** The OCTT responds with a **LogStatusNotificationResponse**
	6. The Charging Station responds with a **TriggerMessageResponse**	**5.** The OCTT sends a **TriggerMessageRequest** With requestedMessage *LogStatusNotification*
	7. The Charging Station sends a **LogStatusNotificationRequest**	**8.** The OCTT responds with a **LogStatusNotificationResponse**
Tool validations	* Step 2: Message: **GetLogResponse** - **status** must be *Accepted* * Step 3: Message: **LogStatusNotificationRequest** - **status** must be *Uploading* * Step 6: Message: **TriggerMessageResponse** - **status** must be *Accepted* * Step 7: Message: **LogStatusNotificationRequest** - **status** must be *Uploading*	
	Post scenario validations: N/a	

Table 190. Test Case Id: TC_F_17_CS

Test case name	**Trigger message - FirmwareStatusNotification - Specific EVSE not relevant**	
Test case Id	TC_F_17_CS	
Use case Id(s)	F06	
Requirement(s)	F06.FR.03,F06.FR.04,F06.FR.05,F06.FR.10	
System under test	Charging Station	
Description	The CSMS can request a Charging Station to send Charging Station-initiated messages. In the request the CSMS indicates which message it wishes to receive.	
Purpose	To verify if the Charging Station is able to send a FirmwareStatusNotificationRequest, after receiving a TriggerMessageRequest even when the CSMS an evseId which is not relevant for the requestedMessage FirmwareStatusNotification.	
Prerequisite(s)	The Charging Station supports sending FirmwareStatusNotifications triggered by a TriggerMessageRequest.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **TriggerMessageResponse**	**1.** The OCTT sends a **TriggerMessageRequest** With **requestedMessage** *FirmwareStatusNotification* **evse.id** is *<Configured evseId>*
	3. The Charging Station sends a **FirmwareStatusNotificationRequest**	**4.** The OCTT responds with a **FirmwareStatusNotificationResponse**
Tool validations	* Step 2: Message: **TriggerMessageResponse** - **status** must be *Accepted* * Step 3: Message: **FirmwareStatusNotificationRequest** - **status** must be *Idle*	
	Post scenario validations: N/a	

Table 191. Test Case Id: TC_F_18_CS

Test case name	**Trigger message - FirmwareStatusNotification - Idle**	
Test case Id	TC_F_18_CS	
Use case Id(s)	F06	
Requirement(s)	F06.FR.04,F06.FR.05,F06.FR.10,F06.FR.16,L01.FR.25	
System under test	Charging Station	
Description	The CSMS can request a Charging Station to send Charging Station-initiated messages. In the request the CSMS indicates which message it wishes to receive.	
Purpose	To verify if the Charging Station is able to send a FirmwareStatusNotificationRequest with status Idle, after receiving a TriggerMessageRequest while NOT downloading a firmware file.	
Prerequisite(s)	The Charging Station supports sending FirmwareStatusNotifications triggered by a TriggerMessageRequest.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **TriggerMessageResponse**	**1.** The OCTT sends a **TriggerMessageRequest** With **requestedMessage** *FirmwareStatusNotification*
	3. The Charging Station sends a **FirmwareStatusNotificationRequest**	**4.** The OCTT responds with a **FirmwareStatusNotificationResponse**
Tool validations	* Step 2: Message: **TriggerMessageResponse** - **status** must be *Accepted* * Step 3: Message: **FirmwareStatusNotificationRequest** - **status** must be *Idle*	
	Post scenario validations: N/a	

Table 192. Test Case Id: TC_F_19_CS

Test case name	**Trigger message - FirmwareStatusNotification - Downloading**	
Test case Id	TC_F_19_CS	
Use case Id(s)	F06	
Requirement(s)	F06.FR.04,F06.FR.05,F06.FR.10,L01.FR.26	
System under test	Charging Station	
Description	The CSMS can request a Charging Station to send Charging Station-initiated messages. In the request the CSMS indicates which message it wishes to receive.	
Purpose	To verify if the Charging Station is able to send a FirmwareStatusNotificationRequest with status Downloading, after receiving a TriggerMessageRequest while downloading a firmware file.	
Prerequisite(s)	The Charging Station supports sending FirmwareStatusNotifications triggered by a TriggerMessageRequest.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **UpdateFirmwareResponse**	**1.** The OCTT sends a **UpdateFirmwareRequest** **firmware.location** is *<Configured firmware_location>* **firmware.retrieveDateTime** is *<Current dateTime - 2 hours>* **firmware.installDateTime** is omitted **firmware.signingCertificate** is *<Configured signingCertificate>* **firmware.signature** is *<Configured signature>*
	3. The Charging Station sends a **FirmwareStatusNotificationRequest**	**4.** The OCTT responds with a **FirmwareStatusNotificationResponse**
	6. The Charging Station responds with a **TriggerMessageResponse**	**5.** The OCTT sends a **TriggerMessageRequest** With requestedMessage *FirmwareStatusNotification*
	7. The Charging Station sends a **FirmwareStatusNotificationRequest**	**8.** The OCTT responds with a **FirmwareStatusNotificationResponse**
Tool validations	* Step 2: Message: **UpdateFirmwareResponse** - **status** must be *Accepted* * Step 3: Message: **FirmwareStatusNotificationRequest** - **status** must be *Downloading* * Step 6: Message: **TriggerMessageResponse** - **status** must be *Accepted* * Step 7: Message: **FirmwareStatusNotificationRequest** - **status** must be *Downloading*	
	Post scenario validations: N/a	

Table 193. Test Case Id: TC_F_20_CS

Test case name	**Trigger message - Heartbeat**	
Test case Id	TC_F_20_CS	
Use case Id(s)	F06	
Requirement(s)	F06.FR.04,F06.FR.05,F06.FR.10	
System under test	Charging Station	
Description	The CSMS can request a Charging Station to send Charging Station-initiated messages. In the request the CSMS indicates which message it wishes to receive.	
Purpose	To verify if the Charging Station is able to send a HeartbeatRequest, after receiving a TriggerMessageRequest.	
Prerequisite(s)	The Charging Station supports sending Heartbeats triggered by a TriggerMessageRequest.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **TriggerMessageResponse**	**1.** The OCTT sends a **TriggerMessageRequest** With **requestedMessage** *Heartbeat*
	3. The Charging Station sends a **HeartbeatRequest**	**4.** The OCTT responds with a **HeartbeatResponse**
Tool validations	* Step 2: Message: **TriggerMessageResponse** - **status** must be *Accepted*	
	Post scenario validations: N/a	

Table 194. Test Case Id: TC_F_23_CS

Test case name	**Trigger message - StatusNotification - Specific EVSE - Available**	
Test case Id	TC_F_23_CS	
Use case Id(s)	F06	
Requirement(s)	F06.FR.02,F06.FR.04,F06.FR.05,F06.FR.10	
System under test	Charging Station	
Description	The CSMS can request a Charging Station to send Charging Station-initiated messages. In the request the CSMS indicates which message it wishes to receive.	
Purpose	To verify if the Charging Station is able to send a StatusNotificationRequest message for a specific available EVSE/Connector, after receiving a TriggerMessageRequest message.	
Prerequisite(s)	The Charging Station supports sending StatusNotifications triggered by a TriggerMessageRequest.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **TriggerMessageResponse**	**1.** The OCTT sends a **TriggerMessageRequest** With **requestedMessage** *StatusNotification* **evse.id** *<Configured evseId>* **evse.connectorId** *<Configured connectorId>*
	3. The Charging Station notifies the CSMS about the current state of the connector.	**4.** The OCTT responds accordingly.
Tool validations	* Step 2: Message: **TriggerMessageResponse** - **status** must be *Accepted* * Step 3: Message: **StatusNotificationRequest** - **connectorStatus** *Available* Message: **NotifyEventRequest** - **eventData[0].trigger** *Delta* - **eventData[0].actualValue** *"Available"* - **eventData[0].component.name** *"Connector"* - **eventData[0].variable.name** *"AvailabilityState"*	
	Post scenario validations: N/a	

Table 195. Test Case Id: TC_F_24_CS

Test case name	**Trigger message - StatusNotification - Specific EVSE - Occupied**	
Test case Id	TC_F_24_CS	
Use case Id(s)	F06	
Requirement(s)	F06.FR.02,F06.FR.04,F06.FR.05,F06.FR.10	
System under test	Charging Station	
Description	The CSMS can request a Charging Station to send Charging Station-initiated messages. In the request the CSMS indicates which message it wishes to receive.	
Purpose	To verify if the Charging Station is able to send a StatusNotificationRequest message for a specific occupied EVSE/Connector, after receiving a TriggerMessageRequest message.	
Prerequisite(s)	The Charging Station supports sending StatusNotifications triggered by a TriggerMessageRequest.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): **State is** *EVConnectedPreSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **TriggerMessageResponse**	**1.** The OCTT sends a **TriggerMessageRequest** With **requestedMessage** *StatusNotification* **evse.id** *<Configured evseId>* **evse.connectorId** *<Configured connectorId>*
	3. The Charging Station notifies the CSMS about the current state of the connector.	**4.** The OCTT responds accordingly.
Tool validations	* Step 2: Message: **TriggerMessageResponse** - **status** must be *Accepted* * Step 3: Message: **StatusNotificationRequest** - **connectorStatus** *Occupied* Message: **NotifyEventRequest** - **eventData[0].trigger** *Delta* - **eventData[0].actualValue** *"Occupied"* - **eventData[0].component.name** *"Connector"* - **eventData[0].variable.name** *"AvailabilityState"*	
	Post scenario validations: N/a	

Table 196. Test Case Id: TC_F_26_CS

Test case name	**Trigger message - BootNotification - Rejected**	
Test case Id	TC_F_26_CS	
Use case Id(s)	F06	
Requirement(s)	F06.FR.04,F06.FR.05,F06.FR.17	
System under test	Charging Station	
Description	The CSMS can request a Charging Station to send Charging Station-initiated messages. In the request the CSMS indicates which message it wishes to receive.	
Purpose	To verify if the Charging Station rejects resending a BootNotificationRequest, when it has already received an accepted on a previously sent BootNotification, after receiving a TriggerMessageRequest.	
Prerequisite(s)	The Charging Station supports sending BootNotification triggered by a TriggerMessageRequest.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **TriggerMessageResponse**	**1.** The OCTT sends a **TriggerMessageRequest** With **requestedMessage** *BootNotification*
Tool validations	* Step 2: Message: **TriggerMessageResponse** - **status** must be *Rejected*	
	Post scenario validations: N/a	

Table 197. Test Case Id: TC_F_27_CS

Test case name	**Trigger message - NotImplemented**	
Test case Id	TC_F_27_CS	
Use case Id(s)	F06	
Requirement(s)	F06.FR.08	
System under test	Charging Station	
Description	The CSMS can request a Charging Station to send Charging Station-initiated messages. In the request the CSMS indicates which message it wishes to receive.	
Purpose	To verify if the Charging Station is able to report it has not implemented sending a SignCombinedCertificateRequest, after receiving a TriggerMessageRequest.	
Prerequisite(s)	The Charging Station does NOT support sending SignCombinedCertificates triggered by a TriggerMessageRequest.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **TriggerMessageResponse**	**1.** The OCTT sends a **TriggerMessageRequest** With requestedMessage *SignCombinedCertificate*
Tool validations	* Step 2: Message: **TriggerMessageResponse** - **status** must be *NotImplemented*	
	Post scenario validations: N/a	

2.8. G Availability

Table 198. Test Case Id: TC_G_01_CS

Test case name	**Connector status Notification - Available to Occupied**	
Test case Id	TC_G_01_CS	
Use case Id(s)	G01, N07	
Requirement(s)	G01.FR.01, N07.FR.19	
System under test	Charging Station	
Description	A Charging Station sends a notification to the CSMS to inform the CSMS about a Connector status change. This can be done in two ways. Via a StatusNotificationRequest or a NotifyEventRequest from the device model.	
Purpose	To verify whether the Charging Station is able to report that its connector is *Occupied*.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *EVConnectedPreSession*	
Tool validations	N/a	
	Post scenario validations: N/a	

Table 199. Test Case Id: TC_G_02_CS

Test case name	**Connector status Notification - Occupied to Available**	
Test case Id	TC_G_02_CS	
Use case Id(s)	G01, N07	
Requirement(s)	G01.FR.01, N07.FR.19	
System under test	Charging Station	
Description	A Charging Station sends a notification to the CSMS to inform the CSMS about a Connector status change. This can be done in two ways. Via a StatusNotificationRequest or a NotifyEventRequest from the device model.	
Purpose	To verify whether the Charging Station is able to report that its connector is Available_.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): **State is** *EVConnectedPreSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Disconnect the EV and EVSE.*	
	3. The Charging Station notifies the CSMS about the current state of the connector.	**4.** The OCTT responds accordingly.
Tool validations	* Step 3: Message: **StatusNotificationRequest** - **connectorStatus** *Available* Message: **NotifyEventRequest** - **eventData[0].trigger** *Delta* - **eventData[0].actualValue** *"Available"* - **eventData[0].component.name** *"Connector"* - **eventData[0].variable.name** *"AvailabilityState"*	
	Post scenario validations: N/a	

Table 200. Test Case Id: TC_G_03_CS

Test case name	**Change Availability EVSE - Operative to inoperative**	
Test case Id	TC_G_03_CS	
Use case Id(s)	G03	
Requirement(s)	G03.FR.01, G03.FR.02, G03.FR.04, G03.FR.06	
System under test	Charging Station	
Description	This test case covers how the CSMS requests the Charging Station to change the availability of one of the EVSEs from Operative to Inoperative. An EVSE is considered Operative in any status other than Faulted and Unavailable.	
Purpose	To verify if the Charging Station is able to perform the change availability mechanism as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *Unavailable* for *<Configured evseId>*	
Tool validations	N/a	
	Post scenario validations: - A message to report the state of a connector has been received for all connectors belonging to the specified EVSE.	

Table 201. Test Case Id: TC_G_04_CS

Test case name	**Change Availability EVSE - Inoperative to operative**	
Test case Id	TC_G_04_CS	
Use case Id(s)	G03	
Requirement(s)	G03.FR.01, G03.FR.02, G03.FR.04, G03.FR.07	
System under test	Charging Station	
Description	This test case covers how the CSMS requests the Charging Station to change the availability of one of the EVSEs from Inoperative to Operative. An EVSE is considered Operative in any status other than Faulted and Unavailable.	
Purpose	To verify if the Charging Station is able to perform the change availability mechanism as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: *Unavailable* for *<Configured evseId>*	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **ChangeAvailabilityResponse**	**1.** The OCTT sends a **ChangeAvailabilityRequest** with **operationalStatus** *Operative* and **evse.id** *<Configured evseId>*
	3. The Charging Station notifies the CSMS about the current state of all connectors belonging to the specified EVSE (and optionally also from the EVSE itself).	**4.** The OCTT responds accordingly.
Tool validations	* Step 2: Message **ChangeAvailabilityResponse** - **status** *Accepted* * Step 3: Message: **StatusNotificationRequest** - **connectorStatus** *Available* - **evseId** *<Configured evseId>* Message: **NotifyEventRequest** - **eventData[0].trigger** *Delta* - **eventData[0].actualValue** *"Available"* - **eventData[0].component.name** *"EVSE" / Connector* - **eventData[0].component.evse.id** *<Configured evseId>* - **eventData[0].variable.name** *"AvailabilityState"*	
	Post scenario validations: - A message to report the state of a connector has been received for all connectors belonging to the specified EVSE.	

Table 202. Test Case Id: TC_G_05_CS

Test case name	**Change Availability Charging Station - Operative to inoperative**	
Test case Id	TC_G_05_CS	
Use case Id(s)	G04	
Requirement(s)	G04.FR.01, G04.FR.02, G04.FR.03, G04.FR.05, G04.FR.07	
System under test	Charging Station	
Description	This test case describes how the CSMS requests the Charging Station to change the availability from operative to inoperative. A Charging Station is considered Operative when it is charging or ready for charging. A Charging Station is considered Inoperative when it does not allow any charging.	
Purpose	To verify if the Charging Station is able to perform the change availability mechanism as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *Unavailable*	
Tool validations	N/a	
	Post scenario validations: - A message to report the state of a connector has been received for all connectors.	

Table 203. Test Case Id: TC_G_06_CS

Test case name	**Change Availability Charging Station - Inoperative to operative**	
Test case Id	TC_G_06_CS	
Use case Id(s)	G04	
Requirement(s)	G04.FR.01, G04.FR.02, G04.FR.03, G04.FR.05, G04.FR.08	
System under test	Charging Station	
Description	This test case describes how the CSMS requests the Charging Station to change the availability from inperative to operative. A Charging Station is considered Operative when it is charging or ready for charging. A Charging Station is considered Inoperative when it does not allow any charging.	
Purpose	To verify if the Charging Station is able to perform the change availability mechanism as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): State is *Unavailable*	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **ChangeAvailabilityResponse**	**1.** The OCTT sends a **ChangeAvailabilityRequest** with **operationalStatus** *Operative*
	3. The Charging Station notifies the CSMS about the current state of all its connectors (and optionally also the Charging Station itself and all EVSE).	**4.** The OCTT responds accordingly.
Tool validations	* Step 2: Message **ChangeAvailabilityResponse** - **status** *Accepted* * Step 3: Message: **StatusNotificationRequest** - **connectorStatus** *Available* Message: **NotifyEventRequest** - **eventData[0].trigger** *Delta* - **eventData[0].actualValue** *"Available"* - **eventData[0].component.name** *"ChargingStation" / EVSE / Connector* - **eventData[0].variable.name** *"AvailabilityState"*	
	Post scenario validations: - A message to report the state of a connector has been received for all connectors.	

Table 204. Test Case Id: TC_G_07_CS

Test case name	**Change Availability Connector - Operative to inoperative**	
Test case Id	TC_G_07_CS	
Use case Id(s)	G03	
Requirement(s)	G03.FR.01, G03.FR.02, G03.FR.04, G03.FR.06	
System under test	Charging Station	
Description	This test case covers how the CSMS requests the Charging Station to change the availability of one of the Connectors of one EVSE from Operative to Inoperative. A Connector is considered Operative in any status other than Faulted and Unavailable.	
Purpose	To verify if the Charging Station is able to perform the change availability mechanism as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *Unavailable* for *<Configured connectorId>*	
Tool validations	N/a	
	Post scenario validations: - A message to report the state of the connector has been received.	

Table 205. Test Case Id: TC_G_08_CS

Test case name	**Change Availability Connector - Inoperative to operative**	
Test case Id	TC_G_08_CS	
Use case Id(s)	G03	
Requirement(s)	G03.FR.01, G03.FR.02, G03.FR.04, G03.FR.07	
System under test	Charging Station	
Description	This test case covers how the CSMS requests the Charging Station to change the availability of one of the Connectors from one EVSE from Inoperative to Operative. A Connector is considered Operative in any status other than Faulted and Unavailable.	
Purpose	To verify if the Charging Station is able to perform the change availability mechanism as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: *Unavailable* for *<Configured connectorId>*	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **ChangeAvailabilityResponse**	**1.** The OCTT sends a **ChangeAvailabilityRequest** with **operationalStatus** *Operative* and **evse.id** *<Configured evseId>* and **evse.connectorId** *<Configured connectorId>*
	3. The Charging Station notifies the CSMS about the current state of the connectors.	**4.** The OCTT responds accordingly.
Tool validations	* Step 2: Message **ChangeAvailabilityResponse** - **status** *Accepted* * Step 3: Message: **StatusNotificationRequest** - **connectorStatus** *Available* - **evseId** *<Configured evseId>* - **connectorId** *<Configured connectorId>* Message: **NotifyEventRequest** - **eventData[0].trigger** *Delta* - **eventData[0].actualValue** *"Available"* - **eventData[0].component.name** *"Connector"* - **eventData[0].component.evse.id** *<Configured evseId>* - **eventData[0].component.evse.connectorId** *<Configured connectorId>* - **eventData[0].variable.name** *"AvailabilityState"*	
	Post scenario validations: - A message to report the state of the connector has been received.	

Table 206. Test Case Id: TC_G_09_CS

Test case name	**Change Availability EVSE - Operative to operative**	
Test case Id	TC_G_09_CS	
Use case Id(s)	G03	
Requirement(s)	G03.FR.01, G03.FR.02, G03.FR.03, G03.FR.04	
System under test	Charging Station	
Description	This test case covers how the CSMS requests the Charging Station to change the availability of one of the EVSEs from Operative to Operative. An EVSE is considered Operative in any status other than Faulted and Unavailable.	
Purpose	To verify if the Charging Station is able to perform the change availability from Operative to Operative according to the mechanism as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **ChangeAvailabilityResponse**	**1.** The OCTT sends a **ChangeAvailabilityRequest** with **operationalStatus** *Operative* and **evse.id** *<Configured evseId>*
Tool validations	* Step 2: Message **ChangeAvailabilityResponse** - **status** *Accepted*	
	Post scenario validations: N/a	

Table 207. Test Case Id: TC_G_10_CS

Test case name	**Change Availability EVSE - Inoperative to inoperative**	
Test case Id	TC_G_10_CS	
Use case Id(s)	G03	
Requirement(s)	G03.FR.01, G03.FR.02, G03.FR.03, G03.FR.04	
System under test	Charging Station	
Description	This test case covers how the CSMS requests the Charging Station to change the availability of one of the EVSEs from Inoperative to Inoperative. An EVSE is considered Inoperative in status Faulted and Unavailable.	
Purpose	To verify if the Charging Station is able to perform the change availability from inoperative to inoperative according to the mechanism as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: **State is** *Unavailable* for *<Configured evseId>*	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **ChangeAvailabilityResponse**	**1.** The OCTT sends a **ChangeAvailabilityRequest** with **operationalStatus** *Inoperative* and **evse.id** *<Configured evseId>*
Tool validations	* Step 2: Message **ChangeAvailabilityResponse** - **status** *Accepted*	
	Post scenario validations: - A message to report the state of a connector has been received for all connectors.	

Table 208. Test Case Id: TC_G_11_CS

Test case name	**Change Availability EVSE - With ongoing transaction**	
Test case Id	TC_G_11_CS	
Use case Id(s)	G03	
Requirement(s)	G03.FR.01, G03.FR.02, G03.FR.04, G03.FR.05	
System under test	Charging Station	
Description	This test case covers how the CSMS requests the Charging Station to change the availability of one of the EVSEs from Operative to Inoperative. An EVSE is considered Operative in any status other than Faulted and Unavailable.	
Purpose	To verify if the Charging Station is able to perform the change availability during a transaction according to the mechanism as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: **State is** *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **ChangeAvailabilityResponse**	**1.** The OCTT sends a **ChangeAvailabilityRequest** with **operationalStatus** *Inoperative* and **evse.id** *<Configured evseId>*
	Note(s): *Wait for <Configured Transaction Duration>*	
	3. Execute **Reusable State** *StopAuthorized*	
	4. The Charging Station notifies the CSMS about the current state of the connectors of the configured evse.	**5.** The OCTT responds accordingly.
	6. Execute **Reusable State** *EVConnectedPostSession*	
	7. The Charging Station notifies the CSMS about the current state of the connectors of the configured evse.	**8.** The OCTT responds accordingly.
	9. Execute **Reusable State** *EVDisconnected*	
	10. The Charging Station notifies the CSMS about the current state of the connectors of the configured evse.	**11.** The OCTT responds accordingly.
	12. Execute **Reusable State** *ParkingBayUnoccupied*	
	13. The Charging Station notifies the CSMS about the current state of the connectors of the configured evse.	**14.** The OCTT responds accordingly.
	Note(s): *Steps 4, 5, 7, 8, 10, 11, 13, and 14 will only be executed if the previous step ended the transaction*	
Tool validations	* Step 2: Message **ChangeAvailabilityResponse** - **status** *Scheduled* * Step 4, 7, 10, 13: Message: **StatusNotificationRequest** - **connectorStatus** *Unavailable* - **evseId** *<Configured evseId>* Message: **NotifyEventRequest** - **eventData[0].trigger** *Delta* - **eventData[0].actualValue** *"Unavailable"* - **eventData[0].component.name** *"Connector"* - **eventData[0].component.evse.id** *<Configured evseId>* - **eventData[0].variable.name** *"AvailabilityState"*	
	Post scenario validations: - A message to report the state of a connector has been received for all connectors.	

Table 209. Test Case Id: TC_G_12_CS

Test case name	**Change Availability Charging Station - Operative to operative**	
Test case Id	TC_G_12_CS	
Use case Id(s)	G04	
Requirement(s)	G04.FR.01, G04.FR.02, G04.FR.03, G04.FR.04, G04.FR.05	
System under test	Charging Station	
Description	This test case describes how the CSMS requests the Charging Station to change the availability from inoperative to operative. A Charging Station is considered Operative when it is charging or ready for charging. A Charging Station is considered Inoperative when it does not allow any charging.	
Purpose	To verify if the Charging Station is able to perform the change availability from operative to operative according to the mechanism as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **ChangeAvailabilityResponse**	**1.** The OCTT sends a **ChangeAvailabilityRequest** with **operationalStatus** *Operative*
Tool validations	* Step 2: Message **ChangeAvailabilityResponse** - **status** *Accepted*	
	Post scenario validations: - A message to report the state of a connector has been received for all connectors.	

Table 210. Test Case Id: TC_G_13_CS

Test case name	**Change Availability Charging Station - Inoperative to inoperative**	
Test case Id	TC_G_13_CS	
Use case Id(s)	G04	
Requirement(s)	G04.FR.01, G04.FR.02, G04.FR.03, G04.FR.04, G04.FR.05	
System under test	Charging Station	
Description	This test case describes how the CSMS requests the Charging Station to change the availability from operative to inoperative. A Charging Station is considered Operative when it is charging or ready for charging. A Charging Station is considered Inoperative when it does not allow any charging.	
Purpose	To verify if the Charging Station is able to perform the change availability from Inoperative to Inoperative according to the mechanism as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: **State is** *Unavailable*	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **ChangeAvailabilityResponse**	**1.** The OCTT sends a **ChangeAvailabilityRequest** with **operationalStatus** *Inoperative*
	3. The Charging Station notifies the CSMS about the current state of all connectors.	**4.** The OCTT responds accordingly.
Tool validations	* 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"*	
	Post scenario validations: - A message to report the state of a connector has been received for all connectors.	

Table 211. Test Case Id: TC_G_14_CS

Test case name	**Change Availability Charging Station - With ongoing transaction**	
Test case Id	TC_G_14_CS	
Use case Id(s)	G04	
Requirement(s)	G04.FR.01, G04.FR.02, G04.FR.03, G04.FR.05, G04.FR.06	
System under test	Charging Station	
Description	This test case covers how the CSMS requests the Charging Station to change the availability of one of the EVSEs from Operative to Inoperative. An EVSE is considered Operative in any status other than Faulted and Unavailable.	
Purpose	To verify if the Charging Station is able to perform the change availability during a transaction according to the mechanism as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: **State is** *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **ChangeAvailabilityResponse**	**1.** The OCTT sends a **ChangeAvailabilityRequest** with **operationalStatus** *Inoperative*
	3. The Charging Station notifies the CSMS about the current state of the connectors of the EVSEs that do not have an active transaction.	**4.** The OCTT responds accordingly.
	Note(s): *Wait for <Configured Transaction Duration>*	
	5. Execute **Reusable State** *StopAuthorized*	
	6. The Charging Station notifies the CSMS about the current state of the connectors of the configured evse.	**7.** The OCTT responds accordingly.
	8. Execute **Reusable State** *EVConnectedPostSession*	
	9. The Charging Station notifies the CSMS about the current state of the connectors of the configured evse.	**10.** The OCTT responds accordingly.
	11. Execute **Reusable State** *EVDisconnected*	
	12. The Charging Station notifies the CSMS about the current state of the connectors of the configured evse.	**13.** The OCTT responds accordingly.
	14. Execute **Reusable State** *ParkingBayUnoccupied*	
	15. The Charging Station notifies the CSMS about the current state of the connectors of the configured evse.	**16.** The OCTT responds accordingly.
	Note(s): *Steps 6, 7, 9, 10, 12, 13, 15, and 16 will only be executed if the previous step ended the transaction*	

Test case name	**Change Availability Charging Station - With ongoing transaction**
Tool validations	* Step 2: Message **ChangeAvailabilityResponse** - **status** *Scheduled* * Step 7: Message: **StatusNotificationRequest** - **connectorStatus** *Unavailable* - **evseId** not *0* - **connectorId** not *0* Message: **NotifyEventRequest** - **eventData[0].trigger** *Delta* - **eventData[0].actualValue** *"Unavailable"* - **eventData[0].component.name** *"Connector"* - **eventData[0].variable.name** *"AvailabilityState"*
	Post scenario validations: - A message to report the state of a connector has been received for all connectors.

Table 212. Test Case Id: TC_G_15_CS

Test case name	**Change Availability Connector - Operative to operative**	
Test case Id	TC_G_15_CS	
Use case Id(s)	G03	
Requirement(s)	G03.FR.01, G03.FR.02, G03.FR.03, G03.FR.04	
System under test	Charging Station	
Description	This test case covers how the CSMS requests the Charging Station to change the availability of one of the EVSEs from Operative to Inoperative. An EVSE is considered Operative in any status other than Faulted and Unavailable.	
Purpose	To verify if the Charging Station is able to perform the change availability from Operative to Operative of one connector according to the mechanism as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **ChangeAvailabilityResponse**	**1.** The OCTT sends a **ChangeAvailabilityRequest** with **operationalStatus** *Operative* and **evse.id** *<Configured evseId>* and **evse.connectorId** *<Configured connectorId>*
Tool validations	* Step 2: Message **ChangeAvailabilityResponse** - **status** *Accepted*	
	Post scenario validations: - A message to report the state of a connector has been received for all connectors.	

Table 213. Test Case Id: TC_G_16_CS

Test case name	**Change Availability Connector - Inoperative to inoperative**	
Test case Id	TC_G_16_CS	
Use case Id(s)	G03	
Requirement(s)	G03.FR.01, G03.FR.02, G03.FR.03, G03.FR.04	
System under test	Charging Station	
Description	This test case covers how the CSMS requests the Charging Station to change the availability of one of the EVSEs from Operative to Inoperative. An EVSE is considered Operative in any status other than Faulted and Unavailable.	
Purpose	To verify if the Charging Station is able to perform the change availability from inopperative to inoperative on one connector according to the mechanism as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: **State is** *Unavailable*	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **ChangeAvailabilityResponse**	**1.** The OCTT sends a **ChangeAvailabilityRequest** with **operationalStatus** *Inoperative* and **evse.id** *<Configured evseId>* and **evse.connectorId** *<Configured connectorId>*
Tool validations	* Step 2: Message **ChangeAvailabilityResponse** - **status** *Accepted*	
	Post scenario validations: - A message to report the state of a connector has been received for all connectors.	

Table 214. Test Case Id: TC_G_17_CS

Test case name	**Change Availability Connector - With ongoing transaction**	
Test case Id	TC_G_17_CS	
Use case Id(s)	G03	
Requirement(s)	G03.FR.01, G03.FR.02, G03.FR.04, G03.FR.05	
System under test	Charging Station	
Description	This test case covers how the CSMS requests the Charging Station to change the availability of one of the EVSEs from Operative to Inoperative. An EVSE is considered Operative in any status other than Faulted and Unavailable.	
Purpose	To verify if the Charging Station is able to perform the change availability during a transaction according to the mechanism as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: **State is** *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **ChangeAvailabilityResponse**	**1.** The OCTT sends a **ChangeAvailabilityRequest** with **operationalStatus** *Inoperative* and **evse.id** *<Configured evseId>* and **evse.connectorId** *<Configured connectorId>*
	Note(s): *Wait for <Configured Transaction Duration>*	
	3. Execute **Reusable State** *StopAuthorized*	
	4. The Charging Station notifies the CSMS about the current state of the connectors of the configured evse.	**5.** The OCTT responds accordingly.
	6. Execute **Reusable State** *EVConnectedPostSession*	
	7. The Charging Station notifies the CSMS about the current state of the connectors of the configured evse.	**8.** The OCTT responds accordingly.
	9. Execute **Reusable State** *EVDisconnected*	
	10. The Charging Station notifies the CSMS about the current state of the connectors of the configured evse.	**11.** The OCTT responds accordingly.
	12. Execute **Reusable State** *ParkingBayUnoccupied*	
	13. The Charging Station notifies the CSMS about the current state of the connectors of the configured evse.	**14.** The OCTT responds accordingly.
	Note(s): *Steps 4, 5, 7, 8, 10, 11, 13, and 14 will only be executed if the previous step ended the transaction*	

Test case name	**Change Availability Connector - With ongoing transaction**
Tool validations	* Step 2: Message **ChangeAvailabilityResponse** - **status** *Scheduled* * Step 7: Message: **StatusNotificationRequest** - **connectorStatus** *Unavailable* - **evseId** *<Configured evseId>* - **connectorId** *<Configured connectorId>* Message: **NotifyEventRequest** - **eventData[0].trigger** *Delta* - **eventData[0].actualValue** *"Unavailable"* - **eventData[0].component.name** *"Connector"* - **eventData[0].component.evse** not *omit* - **eventData[0].component.evse.id** *<Configured evseId>* - **eventData[0].component.evse.connectorId** *<Configured connectorId>* - **eventData[0].variable.name** *"AvailabilityState"*
	Post scenario validations: - A message to report the state of a connector has been received for all connectors.

Table 215. Test Case Id: TC_G_18_CS

Test case name	**Change Availability EVSE - state persists across reboot**	
Test case Id	TC_G_18_CS	
Use case Id(s)	G03	
Requirement(s)	G03.FR.08. G01.FR.01	
System under test	Charging Station	
Description	This test case covers how the CSMS requests the Charging Station to change the availability of one of the EVSEs from Operative to Inoperative. An EVSE is considered Operative in any status other than Faulted and Unavailable.	
Purpose	To verify if the Charging Station sets the availability persistent across reboot/power loss as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: **state** is *Unavailable*	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **ChangeAvailabilityResponse**	**1.** The OCTT sends a **ChangeAvailabilityRequest** with **operationalStatus** *Inoperative* AND **evse.id** *<Configured evseId>*
	3. The Charging Station notifies the CSMS about the current state of all connectors.	**4.** The OCTT responds accordingly.
	5. Execute **Reusable State** *Booted* Note(s): - *After booting the charging station should send the following status:* Message: **StatusNotificationRequest** - **connectorStatus** *Unavailable* - **evseId** *<Configured evseId>* Message: **NotifyEventRequest** - **eventData[0].trigger** *Delta* - **eventData[0].actualValue** *"Unavailable"* - **eventData[0].component.name** *"Connector"* - **eventData[0].component.evse.id** *<Configured evseId>* - **eventData[0].variable.name** *"AvailabilityState"*	
Tool validations	* Step 2: Message **ChangeAvailabilityResponse** - **status** *Accepted* * Step 3: Message: **StatusNotificationRequest** - **evseId** not *0* - **connectorId** not *0* - **connectorStatus** *Unavailable* for **evseId** *<Configured evseId>* - **connectorStatus** *Available* for **evseId** not *<Configured evseId>* Message: **NotifyEventRequest** - **eventData[0].actualValue** *Unavailable* for **evseId** *<Configured evseId>* - **eventData[0].actualValue** *Available* for **evseId** not *<Configured evseId>*	
	Post scenario validations: - A message to report the state of a connector has been received for all connectors.	

Table 216. Test Case Id: TC_G_19_CS

Test case name	**Change Availability Connector - state persists across reboot**	
Test case Id	TC_G_19_CS	
Use case Id(s)	G03	
Requirement(s)	G03.FR.08	
System under test	Charging Station	
Description	This test case covers how the CSMS requests the Charging Station to change the availability of one of the Connectors of one EVSE from Operative to Inoperative. A Connector is considered Operative in any status other than Faulted and Unavailable.	
Purpose	To verify if the Charging Station sets the availability persistent across reboot/power loss as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: **state** is *Unavailable*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *Booting*	
	2. The Charging Station sends a **BootNotificationRequest**	**3.** The OCTT responds with a **BootNotificationResponse**.
	4. The Charging Station reports the status of all its connectors.	**5.** The OCTT responds accordingly.
	6. The Charging Station sends a **SecurityEventNotificationRequest**	**7.** The OCTT responds with a **SecurityEventNotificationResponse**
Tool validations	* Step 4: Message: **StatusNotificationRequest** - **evseId** not *0* - **connectorId** not *0* - **connectorStatus** *Unavailable* for **evseId** *<Configured evseId>* and for **connectorId** *<Configured ConnectorId>* - **connectorStatus** *Available* for **evseId** not *<Configured evseId>* and for **connectorId** *<Configured ConnectorId>* Message: **NotifyEventRequest** - **eventData[0].actualValue** *Unavailable* for **evseId** *<Configured evseId>* and for **connectorId** *<Configured ConnectorId>* - **eventData[0].actualValue** *Available* for **evseId** not *<Configured evseId>* and for **connectorId** *<Configured ConnectorId>* * Step 6: Message: **SecurityEventNotificationRequest** - **type** *"StartupOfTheDevice"* or **type** *"ResetOrReboot"*	
	Post scenario validations: - A message to report the state of a connector has been received for all connectors.	

Table 217. Test Case Id: TC_G_20_CS

Test case name	**Connector status Notification - Lock Failure**	
Test case Id	TC_G_20_CS	
Use case Id(s)	G05	
Requirement(s)	G05.FR.01, G05.FR.02	
System under test	Charging Station	
Description	This test case describes how the EV Driver is prevented from starting a charge session at the Charging Station while the Connector is not locked properly.	
Purpose	To verify if the Charging Station does not start charging and notifies the CSMS when a connector is not locked properly as described at the OCPP specification.	
Prerequisite(s)	- Charging Station has the ConnectorPlugRetentionLock component defined in its Device Model. - MonitoringLevel is set to a level that a connector lock event failure will be reported.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: **State is** *EVConnectedPreSession* Note(s): - *Cable should not be fully plugged in so it cannot lock properly*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *Authorized*	
	2. The Charging Station sends a **NotifyEventRequest**	**3 .** The OCTT responds with a **NotifyEventResponse**
Tool validations	* Step 2: Message: **NotifyEventRequest** - **eventData[0].trigger** *Delta* - **eventData[0].component.name** *"ConnectorPlugRetentionLock"* - **eventData[0].variable.name** *"Problem"* - **eventData[0].actualValue** *"true"*	
	Post scenario validations: - The charging station did not start charging	

Table 218. Test Case Id: TC_G_21_CS

Test case name	**Change Availability Charging Station - state persists across reboot**	
Test case Id	TC_G_21_CS	
Use case Id(s)	G04	
Requirement(s)	G04.FR.09	
System under test	Charging Station	
Description	This test case covers how the CSMS requests the Charging Station to change the availability from Operative to Inoperative. An EVSE is considered Operative in any status other than Faulted and Unavailable.	
Purpose	To verify if the Charging Station sets the availability persistent across reboot/power loss as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: **State is** *Unavailable*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *Booting*	
	2. The Charging Station sends a **BootNotificationRequest**.	**3.** The OCTT responds with a **BootNotificationResponse**.
	4. The Charging Station reports the status of all its connectors.	**5.** The OCTT responds accordingly.
	6. The Charging Station sends a **SecurityEventNotificationRequest**	**7.** The OCTT responds with a **SecurityEventNotificationResponse**
Tool validations	* Step 4: Message: **StatusNotificationRequest** - **evseId** not *0* - **connectorId** not *0* - **connectorStatus** *Unavailable* Message: **NotifyEventRequest** - **eventData[0].trigger** *Delta* - **eventData[0].actualValue** *"Unavailable"* - **eventData[0].variable.name** *"AvailabilityState"* * Step 6: Message: **SecurityEventNotificationRequest** - **type** *"StartupOfTheDevice"* or **type** *"ResetOrReboot"*	
	Post scenario validations: - A message to report the state of a connector has been received for all connectors.	

2.9. H Reservation

Table 219. Test Case Id: TC_H_01_CS

Test case name	**Reserve a specific EVSE - Accepted - Valid idToken**	
Test case Id	TC_H_01_CS	
Use case Id(s)	H01(S2), H03	
Requirement(s)	H01.FR.15, H03.FR.01, H03.FR.09, H03.FR.10	
System under test	Charging Station	
Description	The CSMS is able to reserve a specific EVSE for a specific IdToken by sending a ReserveNowRequest containing an evseId.	
Purpose	To verify if the Charging Station is able to reserve a specific EVSE, until the EV Driver with the specified IdToken arrives.	
Prerequisite(s)	The configuration variable ReservationCtrlr.ReservationAvailable is implemented with value *true*	
Before (Preparations)	**Configuration State:** **ReservationCtrlr.ReservationEnabled** is *true* (If implemented)	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *Reserved* for *<Configured evseId>*	
	2. Execute **Reusable State** *Authorized* Note(s): - *<Configured valid idToken fields> are used for the authorization.*	
	3. Execute **Reusable State** *EnergyTransferStarted*	
Tool validations	* Step 2: *After authorization, connector status must change from Reserved to Available* Message: **StatusNotificationRequest** - **evseId** *<configured evseId>* - **connectorId** *<configured connectorId>* - **connectorStatus** must be *Available* Message: **NotifyEventRequest** - **eventData[0].trigger** must be *Delta* - **eventData[0].actualValue** must be *Available* - **eventData[0].component.name** must be *Connector* - **eventData[0].variable.name** must be *AvailabilityState* - **evse.id** *<configured evseId>* - **connector.id** *<configured connectorId>*	
	Post scenario validations: N/a	

Table 220. Test Case Id: TC_H_02_CS

Test case name	**Reserve a specific EVSE - Accepted - Different idToken**	
Test case Id	TC_H_02_CS	
Use case Id(s)	H01(S2), H03	
Requirement(s)	H03.FR.01, F01.FR.22	
System under test	Charging Station	
Description	The CSMS is able to reserve a specific EVSE for a specific IdToken by sending a ReserveNowRequest containing an evseId. Starting a transaction can be done in two ways (this is configurable by the OCTT; A. Using local authorization B. Using a **RequestStartTransactionRequest**	
Purpose	To verify if the Charging Station rejects all idToken, except the one specified for the reserved EVSE. EV is plugged in before authorization to check that station is able to handle this correctly. When TxStartPoint contains EVConnected this triggers starting of a transaction, but charging must not be allowed when idToken does not match reservation.	
Prerequisite(s)	The configuration variable ReservationCtrlr.ReservationAvailable is implemented with value *true*	
Before (Preparations)	**Configuration State:** **ReservationCtrlr.ReservationEnabled** is *true* (If implemented)	
	Memory State: N/a	
	Reusable State(s): **State is** *<Configured evseId>* is *Reserved* for *<Configured valid_idtoken1_idtoken>* **State is** *EVConnectedPreSession*	
Main A (Test scenario)	**Charging Station**	**CSMS**
	Manual action: Authorize with *<Configured valid_idtoken2_idtoken>*.	
	Execute **reusable state** *Authorized*	
	Note(s): *The test is a PASS, if the OCTT does not receive an a* ***TransactionEventRequest*** *with* ***chargingState*** *= Charging within the configured messageTimeout.*	
Tool validations	N/a	
Main B (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **RequestStartTransactionResponse**	**1.** The OCTT sends a **RequestStartTransactionRequest** with **idToken.idToken** *<Configured valid_idtoken2_idtoken>* **idToken.type** *<Configured valid_idtoken2_type>* **evseId** *<Configured evseId>*
Tool validations	* Step 2: Message: **RequestStartTransactionResponse** - **status** must be *Rejected*	
	Post scenario validations: N/a	

Table 221. Test Case Id: TC_H_03_CS

Test case name	**Reserve a specific EVSE - Occupied - EVSE Reserved**	
Test case Id	TC_H_03_CS	
Use case Id(s)	H01(S2)	
Requirement(s)	H01.FR.11	
System under test	Charging Station	
Description	The CSMS is able to reserve a specific EVSE for a specific IdToken by sending a ReserveNowRequest containing an evseId.	
Purpose	To verify if the Charging Station is able to respond with status Occupied, when the requested EVSE is already reserved.	
Prerequisite(s)	The configuration variable ReservationCtrlr.ReservationAvailable is implemented with value *true*	
Before (Preparations)	**Configuration State:** **ReservationCtrlr.ReservationEnabled** is *true* (If implemented)	
	Memory State: *<Configured evseID>* is *Reserved*	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **ReserveNowResponse**	**1.** The OCTT sends a **ReserveNowRequest** with **evseId** is *<Configured evseId>* **idToken.idToken** *<Configured valid_idtoken2_idtoken>* **idToken.type** *<Configured valid_idtoken2_type>*
Tool validations	* Step 2: Message: **ReserveNowResponse** - **status** must be *Occupied*	
	Post scenario validations: N/a	

Table 222. Test Case Id: TC_H_04_CS

Test case name	**Reserve a specific EVSE - Occupied - EVSE Occupied**	
Test case Id	TC_H_04_CS	
Use case Id(s)	H01(S2)	
Requirement(s)	H01.FR.13	
System under test	Charging Station	
Description	The CSMS is able to reserve a specific EVSE for a specific IdToken by sending a ReserveNowRequest containing an evseId.	
Purpose	To verify if the Charging Station is able to respond with status Occupied, when the requested EVSE is occupied.	
Prerequisite(s)	The configuration variable ReservationCtrlr.ReservationAvailable is implemented with value *true*	
Before (Preparations)	**Configuration State:** **ReservationCtrlr.ReservationEnabled** is *true* (If implemented)	
	Memory State: **State is** *EnergyTransferStarted*	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **ReserveNowResponse**	**1.** The OCTT sends a **ReserveNowRequest** with **evseId** is *<Configured evseId>* **idToken.idToken** *<Configured valid_idtoken2_idtoken>* **idToken.type** *<Configured valid_idtoken2_type>*
Tool validations	* Step 2: Message: **ReserveNowResponse** - **status** must be *Occupied*	
	Post scenario validations: N/a	

Table 223. Test Case Id: TC_H_06_CS

Test case name	**Reserve a specific EVSE - Unavailable**	
Test case Id	TC_H_06_CS	
Use case Id(s)	H01(S2)	
Requirement(s)	H01.FR.14	
System under test	Charging Station	
Description	The CSMS is able to reserve a specific EVSE for a specific IdToken by sending a ReserveNowRequest containing an evseId.	
Purpose	To verify if the Charging Station is able to respond with status Unavailable, when the requested EVSE is unavailable.	
Prerequisite(s)	The configuration variable ReservationCtrlr.ReservationAvailable is implemented with value *true*	
Before (Preparations)	**Configuration State:** **ReservationCtrlr.ReservationEnabled** is *true* (If implemented)	
	Memory State: *<Configured evseID>* is *Unavailable*	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **ReserveNowResponse**	**1.** The OCTT sends a **ReserveNowRequest** with **evseId** is *<Configured evseId>* **idToken.idToken** *<Configured valid_idtoken2_idtoken>* **idToken.type** *<Configured valid_idtoken2_type>*
Tool validations	* Step 2: Message: **ReserveNowResponse** - **status** must be *Unavailable*	
	Post scenario validations: N/a	

Table 224. Test Case Id: TC_H_07_CS

Test case name	**Reserve a specific EVSE - Reservation Ended / not used**	
Test case Id	TC_H_07_CS	
Use case Id(s)	H01(S2), H04	
Requirement(s)	H04.FR.01,H04.FR.02,H04.FR.03	
System under test	Charging Station	
Description	The CSMS is able to reserve a specific EVSE for a specific IdToken by sending a ReserveNowRequest containing an evseId.	
Purpose	To verify if the Charging Station is able to end the reservation, when the EV Driver with the specified IdToken arrives, does not arrive before the set **expiryDateTime** is reached.	
Prerequisite(s)	The configuration variable ReservationCtrlr.ReservationAvailable is implemented with value *true*	
Before (Preparations)	**Configuration State:** **ReservationCtrlr.ReservationEnabled** is *true* (If implemented)	
	Memory State: *<Configured evseID>* is *Reserved*	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The Charging Station notifies the CSMS about the status change of the connector. Note(s): - *The OCTT expects that the Charging Station sets the availabilityState of the EVSE and corresponding connectors back to Available after the expiry time of 60 seconds is reached.*	**2.** The OCTT responds accordingly.
	3. The Charging Station sends a **ReservationStatusUpdateRequest**.	**4.** The OCTT responds with a **ReservationStatusUpdateResponse**.
	5. Execute **Reusable State** *Authorized* Note(s): - *<Configured valid idtoken fields2> are used for the authorization.*	
	6. Execute **Reusable State** *EnergyTransferStarted*	

Test case name	**Reserve a specific EVSE - Reservation Ended / not used**
Tool validations	* Step 1: Message: **StatusNotificationRequest** - **connectorStatus** must be *Available* Message: **NotifyEventRequest** - **eventData[0].trigger** must be *Delta* - **eventData[0].actualValue** must be *Available* - **eventData[0].component.name** must be *Connector* - **eventData[0].variable.name** must be *AvailabilityState* (Reporting the AvailabilityState of the EVSE component itself is optional.) Message: **NotifyEventRequest** - **eventData[0].trigger** must be *Delta* - **eventData[0].actualValue** must be *Available* - **eventData[0].component.name** must be *EVSE* - **eventData[0].variable.name** must be *AvailabilityState* * Step 3: Message: **ReservationStatusUpdateRequest** - **reservationId** must be *<Generated reservationId>* - **reservationUpdateStatus** must be *Expired*
	Post scenario validations: N/a

Table 225. Test Case Id: TC_H_08_CS

Test case name	**Reserve an unspecified EVSE - Accepted**	
Test case Id	TC_H_08_CS	
Use case Id(s)	H01(S1), H03	
Requirement(s)	H01.FR.04,H01.FR.07,H01.FR.15,H03.FR.03	
System under test	Charging Station	
Description	The CSMS is able to reserve an unspecified EVSE for a specific IdToken by sending a ReserveNowRequest without an evseId.	
Purpose	To verify if the Charging Station is able to reserve an unspecified EVSE, until the EV Driver with the specified IdToken arrives.	
Prerequisite(s)	- The configuration variable ReservationCtrlr.ReservationAvailable is implemented with value *true* - The Charging Station has the configuration variable **ReservationNonEvseSpecific** implemented with value *true*	
Before (Preparations)	**Configuration State:** **ReservationCtrlr.ReservationEnabled** is *true* (If implemented)	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **ReserveNowResponse**	**1.** The OCTT sends a **ReserveNowRequest** with **evseId** is Omitted **idToken.idToken** *<Configured valid_idtoken_idtoken>* **idToken.type** *<Configured valid_idtoken_type>*
	3. The Charging Station notifies the CSMS about the status change of the connector. Note(s): *- If the Charging Station has only one EVSE, it sets the availabilityState of the EVSE and corresponding connectors to Reserved.* *- Reporting the AvailabilityState of the EVSE component itself is optional.*	**4.** The OCTT responds accordingly.
	3. Execute **Reusable State** *Authorized* Note(s): *- <Configured valid idToken2 fields> are used for the authorization.*	
	4. Execute **Reusable State** *EnergyTransferStarted*	

Test case name	**Reserve an unspecified EVSE - Accepted**
Tool validations	* Step 2: Message: **ReserveNowResponse** - **status** must be *Accepted* * Step 3: Message: **StatusNotificationRequest** - **connectorStatus** must be *Reserved* - **evseId** must be *<Configured evseId>* - **connectorId** must be *<Configured connectorId>* Message: **NotifyEventRequest** - **trigger** must be *Delta* - **actualValue** must be *"Reserved"* - **component.name** must be *"Connector"* - **evse.id** must be *<Configured evseId>* - **eves.connectorId** must be *<Configured connectorId>* - **variable.name** must be *"AvailabilityState"* (Optional) Message: **NotifyEventRequest** - **eventData[0].trigger** must be *Delta* - **eventData[0].actualValue** must be *Available* - **eventData[0].component.name** must be *EVSE* - **eventData[0].variable.name** must be *AvailabilityState*
	Post scenario validations: N/a

Table 226. Test Case Id: TC_H_09_CS

Test case name	**Reserve an unspecified EVSE - Occupied - EVSE Reserved**	
Test case Id	TC_H_09_CS	
Use case Id(s)	H01(S1)	
Requirement(s)	H01.FR.11	
System under test	Charging Station	
Description	The CSMS is able to reserve an unspecified EVSE for a specific IdToken by sending a ReserveNowRequest without an evseId.	
Purpose	To verify if the Charging Station is able to respond with status Occupied, when all EVSE are already reserved.	
Prerequisite(s)	- The configuration variable ReservationCtrlr.ReservationAvailable is implemented with value *true* - The Charging Station has the configuration variable **ReservationNonEvseSpecific** implemented with value *true*	
Before (Preparations)	**Configuration State:** **ReservationCtrlr.ReservationEnabled** is *true* (If implemented)	
	Memory State: N/a	
	Reusable State(s): All EVSE are *Reserved*	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **ReserveNowResponse**	**1.** The OCTT sends a **ReserveNowRequest** with **evseId** is Omitted **idToken.idToken** *<Configured valid_idtoken_idtoken>* **idToken.type** *<Configured valid_idtoken_type>*
Tool validations	* Step 2: Message: **ReserveNowResponse** - **status** must be *Occupied*	
	Post scenario validations: N/a	

Table 227. Test Case Id: TC_H_10_CS

Test case name	**Reserve an unspecified EVSE - Occupied - EVSE Occupied**	
Test case Id	TC_H_10_CS	
Use case Id(s)	H01(S1)	
Requirement(s)	H01.FR.13	
System under test	Charging Station	
Description	The CSMS is able to reserve an unspecified EVSE for a specific IdToken by sending a ReserveNowRequest without an evseId.	
Purpose	To verify if the Charging Station is able to respond with status Occupied, when all EVSE are occupied.	
Prerequisite(s)	- The configuration variable ReservationCtrlr.ReservationAvailable is implemented with value *true* - The Charging Station has the configuration variable **ReservationNonEvseSpecific** implemented with value *true*	
Before (Preparations)	**Configuration State:** **ReservationCtrlr.ReservationEnabled** is *true* (If implemented)	
	Memory State: N/a	
	Reusable State(s): **State is** *EnergyTransferStarted* for all EVSE	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **ReserveNowResponse**	**1.** The OCTT sends a **ReserveNowRequest** with **evseId** is Omitted **idToken.idToken** *<Configured valid_idtoken_idtoken>* **idToken.type** *<Configured valid_idtoken_type>* **expiryDateTime** *<Configured expiryDateTime>*
Tool validations	* Step 2: Message: **ReserveNowResponse** - **status** must be *Occupied*	
	Post scenario validations: N/a	

Table 228. Test Case Id: TC_H_12_CS

Test case name	**Reserve an unspecified EVSE - Unavailable**	
Test case Id	TC_H_12_CS	
Use case Id(s)	H01(S1)	
Requirement(s)	H01.FR.14	
System under test	Charging Station	
Description	The CSMS is able to reserve an unspecified EVSE for a specific IdToken by sending a ReserveNowRequest without an evseId.	
Purpose	To verify if the Charging Station is able to respond with status Unavailable, when all EVSE are unavailable.	
Prerequisite(s)	- The configuration variable ReservationCtrlr.ReservationAvailable is implemented with value *true* - The Charging Station has the configuration variable **ReservationNonEvseSpecific** implemented with value *true*	
Before (Preparations)	**Configuration State:** **ReservationCtrlr.ReservationEnabled** is *true* (If implemented)	
	Memory State: Charging Station is *Unavailable*	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **ReserveNowResponse**	**1.** The OCTT sends a **ReserveNowRequest** with **evseId** is Omitted **idToken.idToken** *<Configured valid_idtoken_idtoken>* **idToken.type** *<Configured valid_idtoken_type>*
Tool validations	* Step 2: Message: **ReserveNowResponse** - **status** must be *Unavailable*	
	Post scenario validations: N/a	

Table 229. Test Case Id: TC_H_13_CS

Test case name	**Reserve an unspecified EVSE - Rejected**	
Test case Id	TC_H_13_CS	
Use case Id(s)	H01(S1)	
Requirement(s)	H01.FR.19	
System under test	Charging Station	
Description	The CSMS is able to reserve an unspecified EVSE for a specific IdToken by sending a ReserveNowRequest without an evseId.	
Purpose	To verify if the Charging Station is able to respond with status Rejected, when it does not support reserving an unspecified EVSE.	
Prerequisite(s)	- The configuration variable ReservationCtrlr.ReservationAvailable is implemented with value *true* - The Charging Station does NOT have the configuration variable **ReservationNonEvseSpecific** implemented OR the Charging Station does have it implemented with value *false*	
Before (Preparations)	**Configuration State:** **ReservationCtrlr.ReservationEnabled** is *true* (If implemented)	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **ReserveNowResponse**	**1.** The OCTT sends a **ReserveNowRequest** with **evseId** is Omitted **idToken.idToken** *<Configured valid_idtoken_idtoken>* **idToken.type** *<Configured valid_idtoken_type>*
Tool validations	* Step 2: Message: **ReserveNowResponse** - **status** must be *Rejected*	
	Post scenario validations: N/a	

Table 230. Test Case Id: TC_H_14_CS

Test case name	**Reserve an unspecified EVSE - Amount of EVSEs available equals the amount of reservations**	
Test case Id	TC_H_14_CS	
Use case Id(s)	H01(S1)	
Requirement(s)	H01.FR.20	
System under test	Charging Station	
Description	The CSMS is able to reserve an unspecified EVSE for a specific IdToken by sending a ReserveNowRequest without an evseId.	
Purpose	To verify if the Charging Station is able to set all available EVSE to reserved, when the amount of EVSEs available equals the amount of reservations.	
Prerequisite(s)	- The configuration variable **ReservationCtrlr.ReservationAvailable** is implemented with value *true* - The Charging Station has the configuration variable **ReservationNonEvseSpecific** implemented with value *true*	
Before (Preparations)	**Configuration State:** **ReservationCtrlr.ReservationEnabled** is *true* (If implemented)	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **ReserveNowResponse**	**1.** The OCTT sends a **ReserveNowRequest** with **evseId** is Omitted **idToken** is *<A different idToken for every reservation being made>* Note(s): *- This step will be executed the amount of times equal to the amount of EVSE the Charging Station has.*
	3. The Charging Station notifies the CSMS about the current state of all its connectors (and optionally also the state of all EVSE).	**4.** The OCTT responds accordingly.
Tool validations	* Step 2: Message: **ReserveNowResponse** - **status** must be *Accepted* * Step 3: Message: **StatusNotificationRequest** - **connectorStatus** must be *Reserved* Message: **NotifyEventRequest** - **eventData[0].trigger** must be *Delta* - **eventData[0].actualValue** must be *Reserved* - **eventData[0].component.name** must be *Connector* - **eventData[0].variable.name** must be *AvailabilityState* (Optional) Message: **NotifyEventRequest** - **eventData[0].trigger** must be *Delta* - **eventData[0].actualValue** must be *Reserved* - **eventData[0].component.name** must be *EVSE* - **eventData[0].variable.name** must be *AvailabilityState*	
	Post scenario validations: N/a	

Table 231. Test Case Id: TC_H_15_CS

Test case name	**Reserve a connector with a specific type - Success**	
Test case Id	TC_H_15_CS	
Use case Id(s)	H01(S3), H03	
Requirement(s)	H01.FR.06,H01.FR.09,H01.FR.15,H03.FR.02	
System under test	Charging Station	
Description	The CSMS is able to reserve an EVSE with a connector with a specific type for a specific IdToken by sending a ReserveNowRequest with a connectorType.	
Purpose	To verify if the Charging Station is able to reserve an EVSE with a connector with a specific type, until the EV Driver with the specified IdToken arrives.	
Prerequisite(s)	- The configuration variable ReservationCtrlr.ReservationAvailable is implemented with value *true* - The Charging Station supports the reservation of a specific connector type.	
Before (Preparations)	**Configuration State:** **ReservationCtrlr.ReservationEnabled** is *true* (If implemented)	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **ReserveNowResponse**	**1.** The OCTT sends a **ReserveNowRequest** with **connectorType** is *<Configured connectorType>* **idToken.idToken** *<Configured valid_idtoken_idtoken>* **idToken.type** *<Configured valid_idtoken_type>*
	3. The Charging Station notifies the CSMS about the status change of the connector. Note(s): *- If the Charging Station has only one available connector of the specified connectorType, it sets the availabilityState of the corresponding EVSE and all connectors of the specified type to Reserved.* *AND If the EVSE has more connector(s) with a different connectorType, the Charging Station must set these other connector(s) to Unavailable.* *- Reporting the AvailabilityState of the EVSE component itself is optional.*	**4.** The OCTT responds accordingly.
	5. Execute **Reusable State** *Authorized* Note(s): *- <Configured valid idToken fields> are used for the authorization.*	
	6. Execute **Reusable State** *EnergyTransferStarted*	

Test case name	**Reserve a connector with a specific type - Success**
Tool validations	* Step 2: Message: **ReserveNowResponse** - **status** must be *Accepted* * Step 3: Message: **StatusNotificationRequest** - **connectorStatus** must be *Reserved* - **evseId** must be *<Configured evseId>* - **connectorId** must be *<Configured connectorId>* Message: **NotifyEventRequest** - **trigger** must be *Delta* - **actualValue** must be *"Reserved"* - **component.name** must be *"Connector"* - **evse.id** must be *<Configured evseId>* - **eves.connectorId** must be *<Configured connectorId>* - **variable.name** must be *"AvailabilityState"* (Optional) Message: **NotifyEventRequest** - **eventData[0].trigger** must be *Delta* - **eventData[0].actualValue** must be *Available* - **eventData[0].component.name** must be *EVSE* - **eventData[0].variable.name** must be *AvailabilityState*
	Post scenario validations: N/a

Table 232. Test Case Id: TC_H_16_CS

Test case name	**Reserve a connector with a specific type - Amount of available connectors of a type equals the amount of reservations**	
Test case Id	TC_H_16_CS	
Use case Id(s)	H01(S3)	
Requirement(s)	H01.FR.11	
System under test	Charging Station	
Description	The CSMS is able to reserve an EVSE with a connector with a specific type for a specific IdToken by sending a ReserveNowRequest with a connectorType.	
Purpose	To verify if the Charging Station is able to reserve an EVSE with a connector with a specific type, until the EV Driver with the specified IdToken arrives.	
Prerequisite(s)	- The configuration variable ReservationCtrlr.ReservationAvailable is implemented with value *true* - The Charging Station supports the reservation of a specific connector type.	
Before (Preparations)	**Configuration State:** **ReservationCtrlr.ReservationEnabled** is *true* (If implemented) *All EVSEs should be reserved*	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **ReserveNowResponse**	**1.** The OCTT sends a **ReserveNowRequest** with **connectorType** is *<Configured connectorType>* **idToken.idToken** *<Configured valid_idtoken_idtoken>* **idToken.type** *<Configured valid_idtoken_type>*
Tool validations	* Step 2: Message: **ReserveNowResponse** - **status** must be *Occupied*	
	Post scenario validations: N/a	

Table 233. Test Case Id: TC_H_17_CS

Test case name	**Cancel reservation of an EVSE - Success**	
Test case Id	TC_H_17_CS	
Use case Id(s)	H02	
Requirement(s)	H02.FR.02	
System under test	Charging Station	
Description	The CSMS is able to cancel a reservation by sending a **CancelReservationRequest** to the Charging Station.	
Purpose	To verify if the Charging Station is able to cancel a reservation when receiving a **CancelReservationRequest** from the CSMS.	
Prerequisite(s)	The configuration variable ReservationCtrlr.ReservationAvailable is implemented with value *true*	
Before (Preparations)	**Configuration State:** **ReservationCtrlr.ReservationEnabled** is *true* (If implemented)	
	Memory State: *<Configured evseID>* is *Reserved*	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **CancelReservationResponse**	**1.** The OCTT sends a **CancelReservationRequest** with **reservationId** is *<Generated reservationId>*
	3. The Charging Station notifies the CSMS about the status change of the connector.	**4.** The OCTT responds accordingly.
Tool validations	* Step 2: Message: **CancelReservationResponse** - **status** must be *Accepted* * Step 3: Message: **StatusNotificationRequest** - **connectorStatus** must be *Available* - **evseId** must be *<Configured evseId>* - **connectorId** must be *<Configured connectorId>* Message: **NotifyEventRequest** - **trigger** must be *Delta* - **actualValue** must be *"Available"* - **component.name** must be *"Connector"* - **evse.id** must be *<Configured evseId>* - **eves.connectorId** must be *<Configured connectorId>* - **variable.name** must be *"AvailabilityState"*	
	Post scenario validations: N/a	

Table 234. Test Case Id: TC_H_18_CS

Test case name	**Cancel reservation of an EVSE - Rejected**	
Test case Id	TC_H_18_CS	
Use case Id(s)	H02	
Requirement(s)	H02.FR.01	
System under test	Charging Station	
Description	The CSMS is able to cancel a reservation by sending a **CancelReservationRequest** to the Charging Station.	
Purpose	To verify if the Charging Station is able to reject a **CancelReservationRequest**, when there is no matching reservationId.	
Prerequisite(s)	The configuration variable ReservationCtrlr.ReservationAvailable is implemented with value *true*	
Before (Preparations)	**Configuration State:** **ReservationCtrlr.ReservationEnabled** is *true* (If implemented)	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **CancelReservationResponse**	**1.** The OCTT sends a **CancelReservationRequest** with **reservationId** is *1*
Tool validations	* Step 2: Message: **CancelReservationResponse** - **status** must be *Rejected*	
	Post scenario validations: N/a	

Table 235. Test Case Id: TC_H_19_CS

Test case name	**Reserve a specific EVSE - Use a reserved EVSE with GroupId**	
Test case Id	TC_H_19_CS	
Use case Id(s)	H01, H03	
Requirement(s)	H01.FR.15,H03.FR.04,H03.FR.08	
System under test	Charging Station	
Description	The CSMS is able to reserve an EVSE for a specific GroupIdToken by sending a **ReserveNowRequest** containing a **groupIdToken**.	
Purpose	To verify if the Charging Station is able to accept an idToken with the same GroupIdToken as the idToken specified for the reservation.	
Prerequisite(s)	The configuration variable ReservationCtrlr.ReservationAvailable is implemented with value *true*	
Before (Preparations)	**Configuration State:** **ReservationCtrlr.ReservationEnabled** is *true* (If implemented)	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **ReserveNowResponse**	**1.** The OCTT sends a **ReserveNowRequest** with **evseId** is *<Configured evseId>* **idToken.idToken** *<Configured valid_idtoken_idtoken>* **idToken.type** *<Configured valid_idtoken_type>* **groupIdToken.idToken** is *<Configured groupIdToken>*
	3. The Charging Station notifies the CSMS about the status change of the connector. Note(s): *- The OCTT expects that the Charging Station sets the availabilityState of the EVSE and corresponding connectors to Reserved.* *- Reporting the AvailabilityState of the EVSE component itself is optional.*	**4.** The OCTT responds accordingly.
	3. Execute **Reusable State** *Authorized* Note(s): *- <Configured valid idtoken fields2> AND <Configured groupIdToken fields> are used for the authorization.*	
	4. Execute **Reusable State** *EnergyTransferStarted*	

Test case name	**Reserve a specific EVSE - Use a reserved EVSE with GroupId**
Tool validations	* Step 2: Message: **ReserveNowResponse** - **status** must be *Accepted* * Step 3: Message: **StatusNotificationRequest** - **connectorStatus** must be *Reserved* - **evseId** must be *<Configured evseId>* - **connectorId** must be *<Configured connectorId>* Message: **NotifyEventRequest** - **trigger** must be *Delta* - **actualValue** must be *"Reserved"* - **component.name** must be *"Connector"* - **evse.id** must be *<Configured evseId>* - **eves.connectorId** must be *<Configured connectorId>* - **variable.name** must be *"AvailabilityState"* (Optional) Message: **NotifyEventRequest** - **eventData[0].trigger** must be *Delta* - **eventData[0].actualValue** must be *Reserved* - **eventData[0].component.name** must be *EVSE* - **eventData[0].variable.name** must be *AvailabilityState*
	Post scenario validations: N/a

Table 236. Test Case Id: TC_H_21_CS

Test case name	**Charging Station cancels reservation when Unavailable**	
Test case Id	TC_H_21_CS	
Use case Id(s)	H01	
Requirement(s)	H01.FR.17	
System under test	Charging Station	
Description	The Charging Station will cancel reservations, when the EVSE specified for a reservation is set to an inoperative state.	
Purpose	To verify if the Charging Station cancels the reservation, when the availability of the EVSE specified for the reservation is set to *Inoperative*.	
Prerequisite(s)	The configuration variable ReservationCtrlr.ReservationAvailable is implemented with value *true*	
Before (Preparations)	**Configuration State:** **ReservationCtrlr.ReservationEnabled** is *true* (If implemented)	
	Memory State: *<Configured evseID>* is *Reserved*	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **ChangeAvailabilityResponse**	**1.** The OCTT sends a **ChangeAvailabilityRequest** with **operationalStatus** *Inoperative* and **evse.id** *<Configured evseId>*
	3. The Charging Station notifies the CSMS about the status change of the connector. Note(s): *- This step needs to be executed for all connectors of the specified EVSE.* *- Reporting the AvailabilityState of the EVSE itself is optional.*	**4.** The OCTT responds accordingly.
	5. The Charging Station sends a **ReservationStatusUpdateRequest**.	**6.** The OCTT responds with a **ReservationStatusUpdateResponse**.
	8. The Charging Station responds with a **ChangeAvailabilityResponse**	**7.** The OCTT sends a **ChangeAvailabilityRequest** with **operationalStatus** *Operative* and **evse.id** *<Configured evseId>*
	9. The Charging Station notifies the CSMS about the status change of the connector. Note(s): *- This step needs to be executed for all connectors of the specified EVSE.* *- Reporting the AvailabilityState of the EVSE itself is optional.*	**10.** The OCTT responds accordingly.
	11. Execute **Reusable State** *Authorized* Note(s): *- <Configured valid idtoken fields2> are used for the authorization.*	
	12. Execute **Reusable State** *EnergyTransferStarted*	

Test case name	**Charging Station cancels reservation when Unavailable**
Tool validations	* Step 2: Message **ChangeAvailabilityResponse** - **status** *Accepted* * Step 3: Message: **StatusNotificationRequest** - **connectorStatus** must be *Unavailable* Message: **NotifyEventRequest** - **eventData[0].trigger** must be *Delta* - **eventData[0].actualValue** must be *Unavailable* - **eventData[0].component.name** must be *Connector* - **eventData[0].variable.name** must be *AvailabilityState* (Reporting the AvailabilityState of the EVSE component itself is optional.) Message: **NotifyEventRequest** - **eventData[0].trigger** must be *Delta* - **eventData[0].actualValue** must be *Unavailable* - **eventData[0].component.name** must be *EVSE* - **eventData[0].variable.name** must be *AvailabilityState* * Step 5: Message: **ReservationStatusUpdateRequest** - **reservationId** must be *<Generated reservationId>* - **reservationUpdateStatus** must be *Removed* * Step 8: Message **ChangeAvailabilityResponse** - **status** *Accepted* * Step 9: Message: **StatusNotificationRequest** - **connectorStatus** must be *Available* Message: **NotifyEventRequest** - **eventData[0].trigger** must be *Delta* - **eventData[0].actualValue** must be *Available* - **eventData[0].component.name** must be *Connector* - **eventData[0].variable.name** must be *AvailabilityState* (Reporting the AvailabilityState of the EVSE component itself is optional.) Message: **NotifyEventRequest** - **eventData[0].trigger** must be *Delta* - **eventData[0].actualValue** must be *Available* - **eventData[0].component.name** must be *EVSE* - **eventData[0].variable.name** must be *AvailabilityState*
	Post scenario validations: N/a

Table 237. Test Case Id: TC_H_22_CS

Test case name	**Reserve a specific EVSE - Configured to Reject**	
Test case Id	TC_H_22_CS	
Use case Id(s)	H01	
Requirement(s)	H01.FR.01	
System under test	Charging Station	
Description	The CSMS is able to reserve a specific EVSE for a specific IdToken by sending a ReserveNowRequest containing an evseId.	
Purpose	To verify if the Charging Station is able to correctly respond when it is configured not to accept reservations.	
Prerequisite(s)	The configuration variable ReservationCtrlr.ReservationAvailable is implemented with value *false*	
Before (Preparations)	**Configuration State:** **ReservationCtrlr.ReservationEnabled** is *false* (If implemented)	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **ReserveNowResponse**	**1.** The OCTT sends a **ReserveNowRequest**
Tool validations	* Step 2: Message: **ReserveNowResponse** - **status** *Rejected*	
	Post scenario validations: N/a	

Table 238. Test Case Id: TC_H_23_CS

Test case name	**Reserve a specific EVSE - Replace reservation**	
Test case Id	TC_H_23_CS	
Use case Id(s)	H01	
Requirement(s)	H01.FR.02	
System under test	Charging Station	
Description	The CSMS is able to reserve a specific EVSE for a specific IdToken by sending a ReserveNowRequest containing an evseId.	
Purpose	To verify if the Charging Station is able to replace a reservation of a specific EVSE, until the EV Driver with the specified IdToken arrives.	
Prerequisite(s)	The configuration variable ReservationCtrlr.ReservationAvailable is implemented with value *true*	
Before (Preparations)	**Configuration State:** **ReservationCtrlr.ReservationEnabled** is *true* (If implemented)	
	Memory State: A reservation is valid on <Configured evseId> with <Configured valid_idtoken>	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **ReserveNowResponse**	**1.** The OCTT sends a **ReserveNowRequest** with **id** *<Configured reservationId>* **evseId** is *<Configured evseId>* **idToken.idToken** *<Configured valid_idtoken_idtoken2>* **idToken.type** *<Configured valid_idtoken_type2>*
	3. Execute **Reusable State** *Authorized* Note(s): - *<Configured valid idToken2 fields> are used for the authorization.*	
	4. Execute **Reusable State** *EnergyTransferStarted*	
Tool validations	* Step 2: Message: **ReserveNowResponse** - **status** must be *Accepted* * Step 3: Message: **StatusNotificationRequest** - **connectorStatus** must be *Reserved* - **evseId** must be *<Specified evseId>* - **connectorId** must be *<Configured connectorId>* Message: **NotifyEventRequest** - **trigger** must be *Delta* - **actualValue** must be *"Reserved"* - **component.name** must be *"Connector"* - **evse.id** must be *<Specified evseId>* - **eves.connectorId** must be *<Configured connectorId>* - **variable.name** must be *"AvailabilityState"* (Optional) Message: **NotifyEventRequest** - **eventData[0].trigger** must be *Delta* - **eventData[0].actualValue** must be *Reserved* - **eventData[0].component.name** must be *EVSE* - **eventData[0].variable.name** must be *AvailabilityState*	

Table 239. Test Case Id: TC_H_24_CS

Test case name	**Reserve an unspecified EVSE - GroupIdToken**	
Test case Id	TC_H_24_CS	
Use case Id(s)	H03	
Requirement(s)	H03.FR.06	
System under test	Charging Station	
Description	The CSMS is able to reserve a specific EVSE for a specific IdToken by sending a ReserveNowRequest containing an evseId.	
Purpose	To verify if the Charging Station is able to reserve a unspecific EVSE, until the EV Driver with the specified groupIdToken arrives.	
Prerequisite(s)	The configuration variable ReservationCtrlr.ReservationAvailable is implemented with value *true*	
Before (Preparations)	**Configuration State:** **ReservationCtrlr.ReservationEnabled** is *true* (If implemented)	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **ReserveNowResponse**	**1.** The OCTT sends a **ReserveNowRequest** with **idToken.idToken** is *<Configured valid_idtoken>* **groupIdToken.idToken** is *<Configured group_idtoken>*
	3. The Charging Station notifies the CSMS about the status change of the connector. Note(s): *- The OCTT expects that the Charging Station sets the availabilityState of the EVSE and corresponding connectors to Reserved.* *- Reporting the AvailabilityState of the EVSE component itself is optional.*	**4.** The OCTT responds accordingly.
	5. Execute **Reusable State** *Authorized* Note(s): *- <Configured valid_idtoken2> is used for the authorization.*	
	6. Execute **Reusable State** *EnergyTransferStarted*	
Tool validations	* Step 2: Message: **ReserveNowResponse** - **status** must be *Accepted* * Step 3: Message: **StatusNotificationRequest** - **connectorStatus** must be *Reserved* Message: **NotifyEventRequest** - **eventData[0].trigger** must be *Delta* - **eventData[0].actualValue** must be *Reserved* - **eventData[0].component.name** must be *Connector* - **eventData[0].variable.name** must be *AvailabilityState* (Optional) Message: **NotifyEventRequest** - **eventData[0].trigger** must be *Delta* - **eventData[0].actualValue** must be *Reserved* - **eventData[0].component.name** must be *EVSE* - **eventData[0].variable.name** must be *AvailabilityState*	

2.10. I Tariff and Cost

Table 240. Test Case Id: TC_I_01_CS

Test case name	**Show EV Driver running total cost during charging - costUpdatedRequest**	
Test case Id	TC_I_01_CS	
Use case Id(s)	I02	
Requirement(s)	I02.FR.02	
System under test	Charging Station	
Description	While a transaction is ongoing, the driver wants to know how much the running total cost is, updated at a relevant interval.	
Purpose	To verify if the Charging Station is able to correctly display the running total cost as described in the OCPP specification.	
Prerequisite(s)	- The Charging Station supports Tariff Information	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): **State is** *EVConnectedPreSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Present valid idToken*	
	1. The Charging Station sends an **AuthorizeRequest**	**2.** The OCTT responds with an **AuthorizeResponse** with - **idTokenInfo.status** *Accepted* - **idTokenInfo.personalMessage.content** *<Configured Cost>*
	3. The Charging Station sends a **TransactionEventRequest** Note(s): *- This step needs to be executed when **TxStartPoint** contains Authorized OR the transaction already started. So in the case **TxStartPoint** contains ParkingBayOccupancy*	**4.** The OCTT responds with a **TransactionEventResponse** with - **idTokenInfo.status** *Accepted*
	5. Execute **Reusable State** *EnergyTransferStarted*	
	6. The Charging Station sends an **TransactionEventRequest**	**7.** The OCTT responds with an **TransactionEventResponse** with - **updatedPersonalMessage.content** *<Configured Cost>*
	9. The Charging Station responds with a **CostUpdatedResponse**	**8.** The OCTT sends a **CostUpdatedRequest** with - **totalCost** *<Configured Cost2>* - **transactionId** *<Configured transactionId>*
	Note(s): *Step 6, 7, 8, and 9 are repeated n times*	

Test case name	**Show EV Driver running total cost during charging - costUpdatedRequest**
Tool validations	* Step 1: Message **AuthorizeRequest** - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>* * Step 3: Message **TransactionEventRequest** - **triggerReason** *Authorized* - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>* if transaction was already started - **eventType** *Updated* else - **eventType** *Started*
	Post scenario validations: - N/a

Table 241. Test Case Id: TC_I_02_CS

Test case name	**Show EV Driver Final Total Cost After Charging**	
Test case Id	TC_I_02_CS	
Use case Id(s)	I03	
Requirement(s)	I03.FR.01, I03.FR.03	
System under test	Charging Station	
Description	While a transaction is ongoing, the driver wants to know how much the running total cost is, updated at a relevant interval.	
Purpose	To verify if the Charging Station is able to correctly display the total cost as described in the OCPP specification.	
Prerequisite(s)	- The Charging Station supports Tariff Information	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): **State is** *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Present valid idToken*	
	1. Execute **Reusable State** *StopAuthorized* Note: IF Message **TransactionEventRequest** - **eventType** *Ended* THEN Message **TransactionEventResponse** - **totalCost** <*Generated Cost*>	
	2. Execute **Reusable State** *EVConnectedPostSession* Note: IF Message **TransactionEventRequest** - **eventType** *Ended* THEN Message **TransactionEventResponse** - **totalCost** <*Generated Cost*>	
	3. Execute **Reusable State** *EVDisconnected* Note: IF Message **TransactionEventRequest** - **eventType** *Ended* THEN Message **TransactionEventResponse** - **totalCost** <*Generated Cost*>	
	4. Execute **Reusable State** *ParkingBayUnoccupied* Note: IF Message **TransactionEventRequest** - **eventType** *Ended* THEN Message **TransactionEventResponse** - **totalCost** <*Generated Cost*>	
Tool validations	N/a	
	Post scenario validations: - N/a	

Table 242. Test Case Id: TC_I_07_CS

Test case name	**Show EV Driver running total cost during charging - transactionEventResponse**	
Test case Id	TC_I_07_CS	
Use case Id(s)	I02	
Requirement(s)	I02.FR.02	
System under test	Charging Station	
Description	While a transaction is ongoing, the driver wants to know how much the running total cost is, updated at a relevant interval.	
Purpose	To verify if the Charging Station is able to correctly display the running total cost as described in the OCPP specification.	
Prerequisite(s)	- The Charging Station supports Tariff Information	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): **State is** *EVConnectedPreSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
	<u>Manual Action</u>: *Present valid idToken*	
	1. The Charging Station sends an **AuthorizeRequest**	**2.** The OCTT responds with an **AuthorizeResponse** with - **idTokenInfo.status** *Accepted* - **idTokenInfo.personalMessage.content** *<Configured Cost>*
	3. The Charging Station sends a **TransactionEventRequest** <u>Note(s)</u>: *- This step needs to be executed when **TxStartPoint** contains Authorized OR the transaction already started. So in the case **TxStartPoint** contains ParkingBayOccupancy*	**4.** The OCTT responds with a **TransactionEventResponse** with - **idTokenInfo.status** *Accepted*
	5. Execute **Reusable State** *EnergyTransferStarted*	
	6. The Charging Station sends an **TransactionEventRequest**	**7.** The OCTT responds with an **TransactionEventResponse** with - **updatedPersonalMessage.content** *<Configured Cost>*
	9. The Charging Station responds with a **TransactionEventResponse**	**8.** The OCTT sends a **TransactionEventRequest** with - **totalCost** *<Configured Cost2>* - **transactionId** *<Configured transactionId>*
	<u>Note(s)</u>: *Step 6, 7, 8, and 9 are repeated n times*	

Test case name	**Show EV Driver running total cost during charging - transactionEventResponse**
Tool validations	* Step 1: Message **AuthorizeRequest** - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>* * Step 3: Message **TransactionEventRequest** - **triggerReason** *Authorized* - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>* if transaction was already started - **eventType** *Updated* else - **eventType** *Started*
	Post scenario validations: - N/a

2.11. J MeterValues

Table 243. Test Case Id: TC_J_01_CS

Test case name	**Clock-aligned Meter Values - No transaction ongoing**	
Test case Id	TC_J_01_CS	
Use case Id(s)	J01	
Requirement(s)	J01.FR.01,J01.FR.02,J01.FR.03,J01.FR.06,J01.FR.07,J01.FR.08,J01.FR.14,J01.FR.15	
System under test	Charging Station	
Description	The Charging Station samples the electrical meter or other sensor/transducer hardware to provide information about its Meter Values. Depending on configuration settings, the Charging Station will send Meter Values.	
Purpose	To verify if the Charging Station is able to send clock-aligned Meter Values, when it is configured to do so.	
Prerequisite(s)	The Charging Station has an energy meter.	
Before (Preparations)	**Configuration State:** **AlignedDataInterval** is *<Configured clock-aligned Meter Values interval>*	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The Charging Station notifies the CSMS about its measured Meter Values. Note(s): *- The Meter Value messages do NOT need to be send/received at the exact specified interval. The configured measurands must be measured at the configured interval.* *- Multiple Meter Value messages may be sent per configured interval. One (or more in case the amount of measured data is too much for one message) for each EVSE and one (or more) for the main power meter (connectorId=0)* *- The OCTT will end the testcase after it has received three Meter Value messages.*	**2.** The OCTT responds accordingly.
Tool validations	* 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">*	
	Post scenario validations: 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.>*	

Table 244. Test Case Id: TC_J_02_CS

Test case name	**Clock-aligned Meter Values - Transaction ongoing**	
Test case Id	TC_J_02_CS	
Use case Id(s)	J01	
Requirement(s)	J01.FR.01,J01.FR.02,J01.FR.03,J01.FR.06,J01.FR.07,J01.FR.08,J01.FR.14,J01.FR.15	
System under test	Charging Station	
Description	The Charging Station samples the electrical meter or other sensor/transducer hardware to provide information about its Meter Values. Depending on configuration settings, the Charging Station will send Meter Values.	
Purpose	To verify if the Charging Station is able to send clock-aligned Meter Values, while a transaction is ongoing, when it is configured to do so.	
Prerequisite(s)	The Charging Station has an energy meter.	
Before (Preparations)	**Configuration State:** **AlignedDataInterval** is *<Configured clock-aligned Meter Values interval>* **AlignedDataSendDuringIdle** is *false* (If implemented)	
	Memory State: N/a	
	Reusable State(s): **State is** *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	Note(s): *- The Charging Station can follow Steps 1 and 2 or Steps 3 and 4*	
	1. The Charging Station notifies the CSMS about its measured Meter Values. Note(s): *- During a transaction the MeterValueRequest can still be used to report meter values for the main power meter (evseId=0) and idle EVSEs* *- The Meter Value messages do NOT need to be send/received at the exact specified interval. The configured measurands must be measured at the configured interval.* *- Multiple Meter Value messages may be sent per configured interval, in case the amount of measured data is too much for one message.*	**2.** The OCTT responds accordingly.
	3. The Charging Station sends a **TransactionEventRequest** Note(s): *- During a transaction the meter values for the configured EVSE with the ongoing transaction should be transmitted using the TransactionEventRequest.* _*- The TransactionEventRequest messages do NOT need to be send/received at the exact specified interval. The configured measurands must be measured at the configured interval.* *- Multiple TransactionEventRequest messages may be sent per configured interval, in case the amount of measured data is too much for one message.* _*- The OCTT will end the testcase after it has the* _*<Configured transaction duration>* is reached._	**4.** The OCTT responds with a **TransactionEventResponse**

Test case name	**Clock-aligned Meter Values - Transaction ongoing**
Tool validations	*Note: The following steps do not need to be sent in a specific order.* * Step 1: Message: **MeterValuesRequest** - **meterValue[0].sampledValue[0].context** must be *Sample.Clock* - **meterValue[0].sampledValue** must contain *<An element per configured measurand at the AlignedDataMeasurands. The measurand field may be omitted when the measurand is "Energy.Active.Import.Register">* * Step 3: Message: **TransactionEventRequest** - **triggerReason** must be *MeterValueClock* - **metervalue[0].sampledValue[0].context** must be *Sample.Clock* - **metervalue[0].sampledValue** must contain *<An element per configured measurand at the AlignedDataMeasurands. The measurand field may be omitted when the measurand is "Energy.Active.Import.Register">*
	Post scenario validations: Message: **TransactionEventRequest** - **timestamp** *<The intervals between the timestamps of the received TransactionEventRequest messages must equal the configured value at AlignedDataInterval. However it is allowed to send multiple Meter Value messages per configured interval, in case the amount of measured data is too much for one message. But the timestamp of these messages must all be the same.>* 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, in case the amount of measured data is too much for one message. But the timestamp of these messages must all be the same.>*

Table 245. Test Case Id: TC_J_03_CS

Test case name	**Clock-aligned Meter Values - EventType Ended**	
Test case Id	TC_J_03_CS	
Use case Id(s)	J01 & (E06,E07,E08,E09,E10,E12)	
Requirement(s)	J01.FR.01,J01.FR.02,J01.FR.03,J01.FR.06,J01.FR.07,J01.FR.08,J01.FR.14,J01.FR.15 & E06.FR.11,E06.FR.17,E07.FR.08,E07.FR.13,E08.FR.09,E09.FR.05,E10.FR.04,E12.FR.07	
System under test	Charging Station	
Description	The Charging Station samples the electrical meter or other sensor/transducer hardware to provide information about its Meter Values. Depending on configuration settings, the Charging Station will send Meter Values.	
Purpose	To verify if the Charging Station is able to send clock-aligned Meter Values when a transaction ends at the TransactionEventRequest with eventType is *Ended*, when it is configured to do so.	
Prerequisite(s)	The Charging Station has an energy meter.	
Before (Preparations)	**Configuration State:** **AlignedDataTxEndedInterval** is *<Configured clock_aligned_tx_ended_meter_values_interval>* **SampledDataTxEndedMeasurands** is *empty string* **AlignedDataSendDuringIdle** is *false* (If implemented)	
	Memory State: N/a	
	Reusable State(s): **State is** *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *ParkingBayUnoccupied* Note(s): - *This step will be executed after the <Configured transaction duration> is reached.* - *This causes the transaction to stop.*	
Tool validations	N/a	
	Post scenario validations: - The **TransactionEventRequest** containing eventType *Ended* contains the MeterValue field. - The MeterValue must contain *<An element per data collection moment indicated by AlignedDataTxEndedInterval. The OCTT will not validate this.>* - **timestamp** *<The intervals between the timestamps of the received Meter Value messages must equal the configured value at AlignedDataTxEndedInterval.>* - **sampledValue[0].context** must be *Sample.Clock* - **sampledValue** must contain *<An element per configured measurand at the AlignedDataTxEndedMeasurands. The measurand field may be omitted when the measurand is "Energy.Active.Import.Register">*	

Table 246. Test Case Id: TC_J_04_CS

Test case name	**Clock-aligned Meter Values - Signed**	
Test case Id	TC_J_04_CS	
Use case Id(s)	J01	
Requirement(s)	J01.FR.21	
System under test	Charging Station	
Description	The Charging Station samples the electrical meter or other sensor/transducer hardware to provide information about its Meter Values. Depending on configuration settings, the Charging Station will send Meter Values.	
Purpose	To verify if the Charging Station is able to send signed clock-aligned Meter Values when a transaction ends at the TransactionEventRequest with eventType is *Ended*, when it is configured to do so.	
Prerequisite(s)	The Charging Station has an energy meter.	
Before (Preparations)	**Configuration State:** **AlignedDataTxEndedInterval** is *<Configured clock_aligned_tx_ended_meter_values_interval>* **AlignedDataSendDuringIdle** is *false* (If implemented) **AlignedDataSignReadings** is *true*	
	Memory State: N/a	
	Reusable State(s): **State is** *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *ParkingBayUnoccupied* Note(s): - *This step will be executed after the <Configured transaction duration> is reached.* - *This causes the transaction to stop.*	
Tool validations	N/a	
	Post scenario validations: - The **TransactionEventRequest** containing eventType *Ended* contains the MeterValue field. - The MeterValue should contain *<An element per data collection moment indicated by AlignedDataTxEndedInterval. The OCTT will not validate this.>* - **timestamp** *<The intervals between the timestamps of the received Meter Value messages should equal the configured value at AlignedDataTxEndedInterval.>* - **sampledValue[0].context** should be *Sample.Clock* - **sampledValue** should contain *<An element per configured measurand at the AlignedDataTxEndedMeasurands. The measurand field may be omitted when the measurand is "Energy.Active.Import.Register">* - **sampledValue.signedMeterValue** should not be omitted - **sampledValue.signedMeterValue.publicKey** should exist and depending on the value of OCPPCommCtrlr.PublicKeyWithSignedMeterValue, should be either "", or a valid public key	

Table 247. Test Case Id: TC_J_06_CS

Test case name	**Clock-aligned Meter Values - No Meter Values during transaction**	
Test case Id	TC_J_06_CS	
Use case Id(s)	J01	
Requirement(s)	N/a	
System under test	Charging Station	
Description	The Charging Station samples the electrical meter or other sensor/transducer hardware to provide information about its Meter Values. Depending on configuration settings, the Charging Station will send Meter Values.	
Purpose	To verify if the Charging Station is able to only send clock-aligned Meter Values when there is no ongoing transaction, when it is configured to do so.	
Prerequisite(s)	- The Charging Station has an energy meter. - The configuration variable AlignedDataSendDuringIdle is implemented.	
Before (Preparations)	**Configuration State:** **AlignedDataInterval** is set to *<Configured clock-aligned Meter Values interval>* **AlignedDataSendDuringIdle** is set to *true*	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The Charging Station notifies the CSMS about its measured Meter Values. Note(s): *- The Meter Value messages do NOT need to be send/received at the exact specified interval. The configured measurands must be measured at the configured interval.* *- Multiple Meter Value messages may be sent per configured interval. One (or more in case the amount of measured data is too much for one message) for each EVSE and one (or more) for the main power meter (evseId=0)*	**2.** The OCTT responds accordingly.
	3. Execute **Reusable State** *EnergyTransferStarted*	
	4. The Charging Station notifies the CSMS about its measured Meter Values. Note(s): _- The Meter Value messages should not be send/received at the exact specified interval.	**5.** The OCTT responds accordingly.
	6. Execute **Reusable State** *ParkingBayUnoccupied* Note(s): *- This step will be executed after the <Configured clock-aligned Meter Values interval + 5 seconds> is reached.*	

Test case name	Clock-aligned Meter Values - No Meter Values during transaction	
	7. The Charging Station notifies the CSMS about its measured Meter Values. Note(s): *- The Meter Value messages do NOT need to be send/received at the exact specified interval. The configured measurands must be measured at the configured interval.* *- Multiple Meter Value messages may be sent per configured interval. One (or more in case the amount of measured data is too much for one message) for each EVSE and one (or more) for the main power meter (evseId=0)*	**8.** The OCTT responds accordingly.
Tool validations	* Step 1 & 7: 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">*	
	Post scenario validations: 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.>* - The Charging Station did not send any message to report Meter Values to the OCTT, during the time the transaction was active at step 3 and 4. This means none of the following; MeterValuesRequest OR TransactionEventRequest containing the MeterValue field.	

Table 248. Test Case Id: TC_J_07_CS

Test case name	**Sampled Meter Values - EventType Started - EVSE known**	
Test case Id	TC_J_07_CS	
Use case Id(s)	J02 & (E01,E02,E03,E09,E04,E05)	
Requirement(s)	J02.FR.01,J02.FR.02,J02.FR.03,J02.FR.04,J02.FR.10,E01.FR.09,E02.FR.09,E03.FR.07,E04.FR.05,E05.FR.05	
System under test	Charging Station	
Description	The Charging Station samples the electrical meter or other sensor/transducer hardware to provide information about its Meter Values. Depending on configuration settings, the Charging Station will send Meter Values.	
Purpose	To verify if the Charging Station is able to send sampled Meter Values when a transaction starts and the EVSE is known, at the TransactionEventRequest with eventType is *Started*, when it is configured to do so.	
Prerequisite(s)	- The Charging Station has an energy meter. - The Charging Station does NOT have the following configuration; TxStartPoint contains *ParkingBayOccupancy*	
Before (Preparations)	**Configuration State:** **TxStartPoint** contains *EVConnected* Note: TxStartPoint contains *EVConnected*, *Authorized*, *PowerPathClosed*, *EnergyTransfer* AND/OR *DataSigned* (At least one of these values must be set).	
	Memory State: N/a	
	Reusable State(s): **State is** *ParkingBayOccupied*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *EVConnectedPreSession*	
	2. Execute **Reusable State** *EnergyTransferStarted*	
Tool validations	N/a	
	Post scenario validations: - The **TransactionEventRequest** containing eventType *Started* contains the MeterValue field. - **sampledValue[0].context** must be *Transaction.Begin* - **sampledValue** must contain *<An element per configured measurand at the SampledDataTxStartedMeasurands. The measurand field may be omitted when the measurand is "Energy.Active.Import.Register">*	

Table 249. Test Case Id: TC_J_08_CS

Test case name	**Sampled Meter Values - Context Transaction.Begin - EVSE not known**	
Test case Id	TC_J_08_CS	
Use case Id(s)	J02 & (E01,E02,E03,E09,E04,E05)	
Requirement(s)	J02.FR.01, J02.FR.02, J02.FR.03, J02.FR.04, J02.FR.10, E01.FR.16, E01.FR.17, E03.FR.11, E04.FR.11, E05.FR.08	
System under test	Charging Station	
Description	The Charging Station samples the electrical meter or other sensor/transducer hardware to provide information about its Meter Values. Depending on configuration settings, the Charging Station will send Meter Values.	
Purpose	To verify if the Charging Station sends Meter Values for Transaction.Begin as soon as the EVSE to be used is known, for a transaction that starts before the cable is plugged in.	
Prerequisite(s)	- The Charging Station has an energy meter. - The Charging Station does NOT have the following configuration; TxStartPoint does NOT contain *ParkingBayOccupancy* OR *Authorized*.	
Before (Preparations)	**Configuration State:** **TxStartPoint** contains *Authorized* Note: TxStartPoint contains *Authorized* AND/OR *ParkingBayOccupancy* (At least one of these values must be set).	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *EnergyTransferStarted*	
Tool validations	N/a	
	Post scenario validations: - The first **TransactionEventRequest** containing a value for **evse**, sent during the execution of reusable state *EVConnectedPreSession* contains the MeterValue field with: - **sampledValue[0].context** must be *Transaction.Begin* - **sampledValue** must contain <*An element per configured measurand at the SampledDataTxStartedMeasurands. The measurand field may be omitted when the measurand is "Energy.Active.Import.Register"*>	

Table 250. Test Case Id: TC_J_09_CS

Test case name	**Sampled Meter Values - EventType Updated**	
Test case Id	TC_J_09_CS	
Use case Id(s)	J02 & (E01,E02,E03,E09,E04,E05)	
Requirement(s)	J02.FR.01, J02.FR.02, J02.FR.03, J02.FR.04, J02.FR.10, J02.FR.11, J02.FR.14, E02.FR.10, E02.FR.11, E03.FR.08, E03.FR.09, E04.FR.06, E04.FR.09, E11.FR.03, E11.FR.06, E12.FR.03, E12.FR.06	
System under test	Charging Station	
Description	The Charging Station samples the electrical meter or other sensor/transducer hardware to provide information about its Meter Values. Depending on configuration settings, the Charging Station will send Meter Values.	
Purpose	To verify if the Charging Station is able to send sampled Meter Values during the transaction, at the TransactionEventRequest with eventType is *Updated*, when it is configured to do so.	
Prerequisite(s)	The Charging Station has an energy meter.	
Before (Preparations)	**Configuration State:** **SampledDataTxUpdatedInterval** is *<Configured sampled Meter Values Updated interval>*	
	Memory State: N/a	
	Reusable State(s): **State is** *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The Charging Station sends a **TransactionEventRequest** Note(s): *- The TransactionEventRequest messages do NOT need to be send/received at the exact specified interval. The configured measurands must be measured at the configured interval.* *- Multiple TransactionEventRequest messages may be sent per configured interval, in case the amount of measured data is too much for one message.* _*- The OCTT will end the testcase after it has the* _*<Configured transaction duration>* is reached._	**2.** The OCTT responds with a **TransactionEventResponse**
Tool validations	* Step 1: Message: **TransactionEventRequest** - **triggerReason** must be *MeterValuePeriodic* - **sampledValue[0].context** must be *Sample.Periodic* - **sampledValue** must contain *<An element per configured measurand at the SampledDataTxUpdatedMeasurands. The measurand field may be omitted when the measurand is "Energy.Active.Import.Register">*	
	Post scenario validations: - **timestamp** *<The intervals between the timestamps of the received TransactionEventRequest messages must equal the configured value at SampledDataTxUpdatedInterval. However it is allowed to send multiple Meter Value messages per configured interval, in case the amount of measured data is too much for one message. But the timestamp of these messages must all be the same.>*	

Table 251. Test Case Id: TC_J_10_CS

Test case name	**Sampled Meter Values - EventType Ended**	
Test case Id	TC_J_10_CS	
Use case Id(s)	J02 & (E06,E07,E08,E09,E10,E12)	
Requirement(s)	J02.FR.01,J02.FR.02,J02.FR.03,J02.FR.04,J02.FR.10,E06.FR.11,E06.FR.17, E07.FR.08,E07.FR.13,E08.FR.09,E09.FR.05,E10.FR.04,E12.FR.07	
System under test	Charging Station	
Description	The Charging Station samples the electrical meter or other sensor/transducer hardware to provide information about its Meter Values. Depending on configuration settings, the Charging Station will send Meter Values.	
Purpose	To verify if the Charging Station is able to send sampled Meter Values when a transaction ends at the TransactionEventRequest with eventType is *Ended*, when it is configured to do so.	
Prerequisite(s)	The Charging Station has an energy meter.	
Before (Preparations)	**Configuration State:** **SampledDataTxEndedInterval** is *<Configured sampled_tx_ended_meter_values_interval>* **AlignedDataTxEndedMeasurands** is *empty string*	
	Memory State: N/a	
	Reusable State(s): **State is** *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *ParkingBayUnoccupied* Note(s): - *This step will be executed after the <Configured transaction duration> is reached.* - *This causes the transaction to stop.*	
Tool validations	N/a	
	Post scenario validations: - The **TransactionEventRequest** containing eventType *Ended* contains the MeterValue field. - The MeterValue must contain *<An element per data collection moment indicated by SampledDataTxEndedInterval. The OCTT will not validate this.>* - **timestamp** *<The intervals between the timestamps of the received Meter Value messages must equal the configured value at SampledDataTxEndedInterval.>* - **sampledValue[0].context** must be *Sample.Periodic* AND one must have *Transaction.End* - **sampledValue** must contain *<An element per configured measurand at the SampledDataTxEndedMeasurands. The measurand field may be omitted when the measurand is "Energy.Active.Import.Register">*	

Table 252. Test Case Id: TC_J_11_CS

Test case name	**Sampled Meter Values - Signed**	
Test case Id	TC_J_11_CS	
Use case Id(s)	J02	
Requirement(s)	J02.FR.21	
System under test	Charging Station	
Description	The Charging Station samples the electrical meter or other sensor/transducer hardware to provide information about its Meter Values. Depending on configuration settings, the Charging Station will send Meter Values.	
Purpose	To verify if the Charging Station is able to send sampled Meter Values when a transaction ends at the TransactionEventRequest with eventType is *Ended*, when it is configured to do so.	
Prerequisite(s)	The Charging Station has an energy meter.	
Before (Preparations)	**Configuration State:** **SampledDataTxEndedInterval** is *<Configured sampled_tx_ended_meter_values_interval>* **SampledDataSignReadings** is true	
	Memory State: N/a	
	Reusable State(s): **State is** *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *ParkingBayUnoccupied* Note(s): *- This step will be executed after the <Configured transaction duration> is reached.* *- This causes the transaction to stop.*	
Tool validations	N/a	
	Post scenario validations: - The **TransactionEventRequest** containing eventType *Ended* contains the MeterValue field. - The MeterValue must contain *<An element per data collection moment indicated by SampledDataTxEndedInterval. The OCTT will not validate this.>* - **timestamp** *<The intervals between the timestamps of the received Meter Value messages must equal the configured value at SampledDataTxEndedInterval.>* - **sampledValue[0].context** must be *Sample.Periodic* AND one must have *Transaction.End* - **sampledValue** must contain *<An element per configured measurand at the SampledDataTxEndedMeasurands. The measurand field may be omitted when the measurand is "Energy.Active.Import.Register">* - **sampledValue.signedMeterValue** should not be omitted - **sampledValue.signedMeterValue.publicKey** should exist and depending on the value of OCPPCommCtrlr.PublicKeyWithSignedMeterValue, should be either "", or a valid public key	

2.12. K SmartCharging

Table 253. Test Case Id: TC_K_01_CS

Test case name	**Set Charging Profile - TxDefaultProfile - Specific EVSE**	
Test case Id	TC_K_01_CS	
Use case Id(s)	K01	
Requirement(s)	K01.FR.07, K01.FR.15	
System under test	Charging Station	
Description	To enable the CSMS to influence the charging power or current drawn from a specific EVSE or the entire Charging Station over a period of time. The CSMS sends a SetChargingProfileRequest to the Charging Station to influence the power or current drawn by EVs. The CSMS calculates a ChargingSchedule to stay within certain limits, which MAY be imposed by any external system.	
Purpose	To verify if the Charging station is able to accept and successfully change to the TxDefaultProfile charging profile sent by the CSMS on a specific EVSE as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SetChargingProfileResponse**	**1.** The OCTT sends a **SetChargingProfileRequest** with **evseId** *<Configured evseId>* AND **chargingProfile.id** *<Configured chargingProfileId>* AND **chargingProfile.chargingProfilePurpose** *TxDefaultProfile* **chargingProfile.chargingSchedule.duration** *<Configured duration>* **chargingProfile.chargingSchedule.chargingRateUnit** *<Configured chargingRateUnit>* **chargingProfile.chargingSchedule.chargingSchedulePeriod.startPeriod** *0* **chargingProfile.chargingSchedule.chargingSchedulePeriod.limit** *if unit is A then 6(A) else 6000(W)* **chargingProfile.chargingSchedule.chargingSchedulePeriod.numberPhases** *<Configured numberPhases>*
	4. The Charging Station responds with a **GetChargingProfilesResponse**	**3.** The OCTT sends a **GetChargingProfilesRequest** with **chargingProfile.chargingProfileId** *<Configured chargingProfileId>* **requestId** *<Generated requestId>*
	5. The Charging Station sends a **ReportChargingProfilesRequest**	**6.** The OCTT responds with a **ReportChargingProfilesResponse**
	Note(s): *- If **tbc** is True at Step 5 then step 5 and 6 will be repeated*	

Test case name	**Set Charging Profile - TxDefaultProfile - Specific EVSE**
Tool validations	* Step 2: Message **SetChargingProfileResponse** - **status** *Accepted* * Step 4: Message **GetChargingProfilesResponse** - **status** *Accepted* * Step 5: Message **ReportChargingProfilesRequest** - **requestId** *<Generated requestId>* - **evseId** *<Configured EVSEId>** - **chargingProfile** *<Configured ChargingProfile>*
	Post scenario validations: - The same profile is reported as send in step 1

Table 254. Test Case Id: TC_K_02_CS

Test case name	**Set Charging Profile - TxProfile without ongoing transaction on the specified EVSE**	
Test case Id	TC_K_02_CS	
Use case Id(s)	K01	
Requirement(s)	K01.FR.04, K01.FR.07, K01.FR.09	
System under test	Charging Station	
Description	To enable the CSMS to influence the charging power or current drawn from a specific EVSE or the entire Charging Station over a period of time. The CSMS sends a SetChargingProfileRequest to the Charging Station to influence the power or current drawn by EVs. The CSMS calculates a ChargingSchedule to stay within certain limits, which MAY be imposed by any external system.	
Purpose	To verify if the Charging station is able to accept and successfully change to the a TxProfile charging profile, without ongoing transaction, sent by the CSMS on a specific EVSE as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SetChargingProfileResponse**	**1.** The OCTT sends a **SetChargingProfileRequest** with **evseId** *<Configured evseId>* AND **chargingProfile.id** *<Configured chargingProfileId>* AND **chargingProfile.chargingProfilePurpose** *TxProfile* AND **chargingProfile.transactionId** *UNKNOWN-TRANSACTION-ID*
Tool validations	* Step 2: Message **SetChargingProfileResponse** - **status** *Rejected*	
	Post scenario validations: - N/a	

Table 255. Test Case Id: TC_K_03_CS

Test case name	**Set Charging Profile - ChargingStationMaxProfile**	
Test case Id	TC_K_03_CS	
Use case Id(s)	K01	
Requirement(s)	N/a	
System under test	Charging Station	
Description	To enable the CSMS to influence the charging power or current drawn from a specific EVSE or the entire Charging Station over a period of time. The CSMS sends a SetChargingProfileRequest to the Charging Station to influence the power or current drawn by EVs. The CSMS calculates a ChargingSchedule to stay within certain limits, which MAY be imposed by any external system.	
Purpose	To verify if the Charging station is able to accept and successfully change to the ChargingStationMaxProfile sent by the CSMS on a specific EVSE as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SetChargingProfileResponse**	**1.** The OCTT sends a **SetChargingProfileRequest** with **chargingProfile.id** *<Configured chargingProfileId>* AND **chargingProfile.chargingProfilePurpose** *ChargingStationMaxProfile* AND **chargingProfile.chargingProfileKind** *Absolute* AND **chargingProfile.chargingSchedule.duration** *<Configured duration>* AND **chargingProfile.chargingSchedule.chargingRateUnit** *<Configured chargingRateUnit>* AND **chargingProfile.chargingSchedule.chargingSchedulePeriod.startPeriod** *0* AND **chargingProfile.chargingSchedule.chargingSchedulePeriod.limit** *if unit is A then 6(A) else 6000(W)* AND **chargingProfile.chargingSchedule.chargingSchedulePeriod.numberPhases** *<Configured numberPhases>* AND **EVSEId** *0*
	4. The Charging Station responds with a **GetChargingProfilesResponse**	**3.** The OCTT sends a **GetChargingProfilesRequest** with **chargingProfile.chargingProfileId** *<Configured chargingProfileId>* **requestId** *<Generated requestId>*
	5. The Charging Station sends a **ReportChargingProfilesRequest**	**6.** The OCTT responds with a **ReportChargingProfilesResponse**
	Note(s): *- If **tbc** is True at Step 5 then step 5 and 6 will be repeated*	

Test case name	**Set Charging Profile - ChargingStationMaxProfile**
Tool validations	* Step 2: Message **SetChargingProfileResponse** - **status** *Accepted* * Step 4: Message **GetChargingProfilesResponse** - **status** *Accepted* * Step 5: Message **ReportChargingProfilesRequest** - **requestId** *<Generated requestId>* - **EvseId** *0* - **chargingProfile** *<Generated chargingProfile>*
	Post scenario validations: - The same profile is reported as send in step 1

Table 256. Test Case Id: TC_K_04_CS

Test case name	**Replace charging profile - With chargingProfileId**	
Test case Id	TC_K_04_CS	
Use case Id(s)	K01	
Requirement(s)	K01.FR.05	
System under test	Charging Station	
Description	To enable the CSMS to influence the charging power or current drawn from a specific EVSE or the entire Charging Station over a period of time. The CSMS sends a SetChargingProfileRequest to the Charging Station to influence the power or current drawn by EVs. The CSMS calculates a ChargingSchedule to stay within certain limits, which MAY be imposed by any external system.	
Purpose	To verify if the Charging station is able to accept and successfully change to the ChargingStationMaxProfile sent by the CSMS on a specific EVSE as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: A chargeprofile with *<Generated chargingProfileId>* AND limit *6.0/6000.0* AND ChargingProfilePurpose TxDefaultProfile is configured	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SetChargingProfileResponse**	**1.** The OCTT sends a **SetChargingProfileRequest** with **chargingProfile.id** *<Configured chargingProfileId>* **chargingProfile.chargingProfilePurpose** *TxDefaultProfile* **chargingProfile.chargingSchedule.chargingSchedulePeriod.limit** *10.0/10000.0*
	4. The Charging Station responds with a **GetChargingProfilesResponse**	**3.** The OCTT sends a **GetChargingProfilesRequest** with **chargingProfile.chargingProfileId** *<Configured chargingProfileId>*
	5. The Charging Station sends a **ReportChargingProfilesRequest**	**6.** The OCTT responds with a **ReportChargingProfilesResponse**
	Note(s): *- If **tbc** is True at Step 5 then step 5 and 6 will be repeated*	
Tool validations	* Step 2: Message **SetChargingProfileResponse** - **status** *Accepted* * Step 4: Message **GetChargingProfilesResponse** - **status** *Accepted* * Step 5: Message **ReportChargingProfilesRequest** - **requestId** *Same Id as in the GetChargingProfilesRequest in step 3* - **EVSEId** *<Configured EVSEId>* - **chargingProfile** *<ChargingProfile set in step 1>*	
	Post scenario validations: - N/a	

Table 257. Test Case Id: TC_K_05_CS

Test case name	**Clear Charging Profile - With chargingProfileId**	
Test case Id	TC_K_05_CS	
Use case Id(s)	K10	
Requirement(s)	K10.FR.03	
System under test	Charging Station	
Description	If the CSMS wishes to clear some or all of the charging profiles that were previously sent to the Charging Station, then the CSMS sends a ClearChargingProfileRequest to the Charging Station.	
Purpose	To verify if the Charging station is able to accept the request and clear a specific charging profile sent with only a chargingProfileId by the CSMS as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: A chargingprofile with *<Configured chargingProfileId>* is configured	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **ClearChargingProfileResponse**	**1.** The OCTT sends a **ClearChargingProfileRequest** with **chargingProfileId** *<Configured chargingProfileId>*
	4. The Charging Station responds with a **GetChargingProfilesResponse**	**3.** The OCTT sends a **GetChargingProfilesRequest** with **chargingProfile.chargingProfileId** *<Configured chargingProfileId>*
Tool validations	* Step 2: Message **ClearChargingProfileResponse** - **status** *Accepted* * Step 4: Message **GetChargingProfilesResponse** - **status** *NoProfiles*	
	Post scenario validations: - N/a	

Table 258. Test Case Id: TC_K_06_CS

Test case name	**Clear Charging Profile - With stackLevel/purpose combination for one profile**	
Test case Id	TC_K_06_CS	
Use case Id(s)	K10	
Requirement(s)	K10.FR.04	
System under test	Charging Station	
Description	If the CSMS wishes to clear some or all of the charging profiles that were previously sent to the Charging Station, then the CSMS sends a ClearChargingProfileRequest to the Charging Station.	
Purpose	To verify if the Charging station is able to accept the request and clear a charging profile sent with a stackLevel/purpose combination by the CSMS as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: A chargingprofile with *<Configured chargingProfilePurpose>* AND *<Configured stackLevel>* is configured	
	Charging State: *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **ClearChargingProfileResponse**	**1.** The OCTT sends a **ClearChargingProfileRequest** with **chargingProfileCriteria.chargingProfilePurpose** *<Configured chargingProfilePurpose>* AND **chargingProfileCriteria.stackLevel** *<Configured stackLevel>*
	4. The Charging Station responds with a **GetChargingProfilesResponse**	**3.** The OCTT sends a **GetChargingProfilesRequest** with **chargingProfile.chargingProfilePurpose** *<Configured chargingProfilePurpose>* AND **chargingProfile.stackLevel** *<Configured stackLevel>*
Tool validations	* Step 2: Message **ClearChargingProfileResponse** - **status** *Accepted* * Step 4: Message **GetChargingProfilesResponse** - **status** *NoProfiles*	
	Post scenario validations: - N/a	

Table 259. Test Case Id: TC_K_07_CS

Test case name	**Clear Charging Profile - With unknown stackLevel/purpose combination**	
Test case Id	TC_K_07_CS	
Use case Id(s)	K10	
Requirement(s)	K10.FR.01	
System under test	Charging Station	
Description	If the CSMS wishes to clear some or all of the charging profiles that were previously sent to the Charging Station, then the CSMS sends a ClearChargingProfileRequest to the Charging Station.	
Purpose	To verify if the Charging station is able to deny the request to clear a specific charging profile when an unknown chargingProfileId and unknown stackLevel/purpose combination is sent by the CSMS as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: A chargingprofile with ChargingProfilePurpose TxDefaultProfile AND StackLevel 1 is configured	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **ClearChargingProfileResponse**	**1.** The OCTT sends a **ClearChargingProfileRequest** with **chargingProfileCriteria.chargingProfilePurpose** *ChargingStationMaxProfile* AND **chargingProfileCriteria.stackLevel** *0*
Tool validations	* Step 2: Message **ClearChargingProfileResponse** - **status** *Unknown*	
	Post scenario validations: - N/a	

Table 260. Test Case Id: TC_K_08_CS

Test case name	**Clear Charging Profile - Without previous charging profile**	
Test case Id	TC_K_08_CS	
Use case Id(s)	K10	
Requirement(s)	K10.FR.01	
System under test	Charging Station	
Description	If the CSMS wishes to clear some or all of the charging profiles that were previously sent to the Charging Station, then the CSMS sends a ClearChargingProfileRequest to the Charging Station.	
Purpose	To verify if the Charging station is able to deny the request to clear a specific charging profile when no charging profiles are configured as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **ClearChargingProfileResponse**	**1.** The OCTT sends a **ClearChargingProfileRequest** with **chargingProfileId** *<Generated chargingProfileId>*
Tool validations	* Step 2: Message **ClearChargingProfileResponse** - **status** *Unknown*	
	Post scenario validations: - N/a	

Table 261. Test Case Id: TC_K_09_CS

Test case name	**Clear Charging Profile - Clearing a TxDefaultProfile - With ongoing transaction**	
Test case Id	TC_K_09_CS	
Use case Id(s)	K10	
Requirement(s)	K10.FR.07	
System under test	Charging Station	
Description	If the CSMS wishes to clear some or all of the charging profiles that were previously sent to the Charging Station, then the CSMS sends a ClearChargingProfileRequest to the Charging Station.	
Purpose	To verify if the Charging station is able to accept the request and clear a TxDefaultProfile by the CSMS as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** **SmartChargingCtrlr.LimitChangeSignificance** is *1.0*	
	Memory State: *SetChargingProfile* with ChargingProfile 1: **chargingProfilePurpose** is *TxDefaultProfile* **chargingProfileKind** should be *Absolute* **stackLevel** should be *0* **evseId** *<Configured evseId>* **validFrom** *<current dateTime - <Configured max time deviation> seconds>* **validTo** *<current dateTime + <Configured max time deviation> + 401 seconds>* **startSchedule** *<current dateTime>* **numberPhases** *<Configured numberPhases>* **ChargingSchedule**: **duration** *400* + <Configured max time deviation> **chargingRateUnit** *<Configured chargingRateUnit>* *Note: If <Configured chargingRateUnit> is W, then the **limit** field will be multiplied by 1000.* **startPeriod** *0*, **limit** *6*	
	Charging State: **State is** *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **GetCompositeScheduleResponse**	**1.** The OCTT sends a **GetCompositeScheduleRequest** with **evseId** is *<Configured evseId>*
	4. The Charging Station responds with a **ClearChargingProfileResponse**	**3.** The OCTT sends a **ClearChargingProfileRequest** with **chargingProfileCriteria.chargingProfilePurpose** *TxDefaultProfile*
	5. The Charging Station responds with a **GetCompositeScheduleResponse**	**6.** The OCTT sends a **GetCompositeScheduleRequest** with **evseId** *<Configured evseId>* **duration** is *300* **chargingRateUnit** *<Configured chargingRateUnit>*

Test case name	**Clear Charging Profile - Clearing a TxDefaultProfile - With ongoing transaction**
Tool validations	* Step 2: (Message: **GetCompositeScheduleResponse**) **status** *Accepted* **evseId** *<Configured evseId>* **ChargingSchedule**: **duration** *300* **chargingRateUnit** *<Configured chargingRateUnit>* *Note: If <Configured chargingRateUnit> is W, then the **limit** field will be multiplied by 1000.* *Note: The period of time between sending the second SetChargingProfileRequest and the **scheduleStart** from the GetCompositeScheduleResponse is called **x**:* **startPeriod** *0*, **limit** *<Local limit of Charging Station (Validation passes if value is 6>* * Step 4: (Message: **ClearChargingProfileResponse**) **status** is *Accepted* * Step 5: (Message: **GetCompositeScheduleResponse**) **status** *Accepted* **evseId** *<Configured evseId>* **ChargingSchedule**: **duration** *300* **chargingRateUnit** *<Configured chargingRateUnit>* *Note: If <Configured chargingRateUnit> is W, then the **limit** field will be multiplied by 1000.* *Note: The period of time between sending the second SetChargingProfileRequest and the **scheduleStart** from the GetCompositeScheduleResponse is called **x**:* **startPeriod** *0*, **limit** *<Local limit of Charging Station (Validation passes if value is NOT 6>*
	Post scenario validations: N/a

Table 262. Test Case Id: TC_K_10_CS

Test case name	**Set Charging Profile - TxDefaultProfile - All EVSE**	
Test case Id	TC_K_10_CS	
Use case Id(s)	K01	
Requirement(s)	K01.FR.07, K01.FR.14	
System under test	Charging Station	
Description	To enable the CSMS to influence the charging power or current drawn from a specific EVSE or the entire Charging Station over a period of time. The CSMS sends a SetChargingProfileRequest to the Charging Station to influence the power or current drawn by EVs. The CSMS calculates a ChargingSchedule to stay within certain limits, which MAY be imposed by any external system.	
Purpose	To verify if the Charging station is able to accept and successfully change to the TxDefaultProfile charging profile sent by the CSMS for all EVSE as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SetChargingProfileResponse**	**1.** The OCTT sends a **SetChargingProfileRequest** with **evseId** *0* AND **chargingProfile.id** *<Configured chargingProfileId>* AND **chargingProfile.chargingProfilePurpose** *TxDefaultProfile* **chargingProfile.chargingSchedule.duration** *<Configured duration>* **chargingProfile.chargingSchedule.chargingRateUnit** *<Configured chargingRateUnit>* **chargingProfile.chargingSchedule.chargingSchedulePeriod.startPeriod** *0* **chargingProfile.chargingSchedule.chargingSchedulePeriod.limit** *6.0* **chargingProfile.chargingSchedule.chargingSchedulePeriod.numberPhases** *<Configured numberPhases>*
	4. The Charging Station responds with a **GetChargingProfilesResponse**	**3.** The OCTT sends a **GetChargingProfilesRequest** with **chargingProfile.chargingProfileId** *<Configured chargingProfileId>* **requestId** *<Generated requestId>*
	5. The Charging Station sends a **ReportChargingProfilesRequest**	**6.** The OCTT responds with a **ReportChargingProfilesResponse**
	Note(s): *- If **tbc** is True at Step 5 then step 5 and 6 will be repeated*	

Test case name	**Set Charging Profile - TxDefaultProfile - All EVSE**
Tool validations	* Step 2: Message **SetChargingProfileResponse** - **status** *Accepted* * Step 4: Message **GetChargingProfilesResponse** - **status** *Accepted* * Step 5: Message **ReportChargingProfilesRequest** - **requestId** *<Generated requestId>* - **EVSEId** *0* - **tbc** *false* - **chargingProfile** *<Configured chargingProfile>*
	Post scenario validations: - The same profile is reported as send in step 1

Table 263. Test Case Id: TC_K_11_CS

Test case name	**Set Charging Profile - Unable to set TxProfile on all EVSE at once**	
Test case Id	TC_K_11_CS	
Use case Id(s)	K01	
Requirement(s)	K01.FR.04, K01.FR.16	
System under test	Charging Station	
Description	To enable the CSMS to influence the charging power or current drawn from a specific EVSE or the entire Charging Station over a period of time. The CSMS sends a SetChargingProfileRequest to the Charging Station to influence the power or current drawn by EVs. The CSMS calculates a ChargingSchedule to stay within certain limits, which MAY be imposed by any external system.	
Purpose	To verify if the Charging station is able to deny a TxProfile when sent to all EVSE as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SetChargingProfileResponse**	**1.** The OCTT sends a **SetChargingProfileRequest** with **evseId** *0* AND **chargingProfile.id** *<Configured chargingProfileId>* AND **chargingProfile.chargingProfilePurpose** *TxProfile*
Tool validations	* Step 2: Message **SetChargingProfileResponse** - **status** *Rejected*	
	Post scenario validations: - N/a	

Table 264. Test Case Id: TC_K_12_CS

Test case name	**Set Charging Profile - ChargerRateUnit Rejected**	
Test case Id	TC_K_12_CS	
Use case Id(s)	K01	
Requirement(s)	K01.FR.26	
System under test	Charging Station	
Description	To enable the CSMS to influence the charging power or current drawn from a specific EVSE or the entire Charging Station over a period of time. The CSMS sends a SetChargingProfileRequest to the Charging Station to influence the power or current drawn by EVs. The CSMS calculates a ChargingSchedule to stay within certain limits, which MAY be imposed by any external system.	
Purpose	To verify if the Charging station is able to deny a chargeProfile when the given ChargerRateUnit is not known by the charger as described at the OCPP specification.	
Prerequisite(s)		
Before (Preparations)	**Configuration State:** This testcase can only be tested when one of the 2 chargingRateUnits is not supported.	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SetChargingProfileResponse**	**1.** The OCTT sends a **SetChargingProfileRequest** with **chargingProfile.id** *<Configured chargingProfileId>* AND **chargingProfile.chargingSchedule.chargingRateUnit** *<Configured chargingRateUnit>*
Tool validations	* Step 2: Message **SetChargingProfileResponse** - **status** *Rejected*	
	Post scenario validations: - N/a	

Table 265. Test Case Id: TC_K_13_CS

Test case name	**Set Charging Profile - Persistent over reboot**	
Test case Id	TC_K_13_CS	
Use case Id(s)	K01	
Requirement(s)	K01.FR.27	
System under test	Charging Station	
Description	To enable the CSMS to influence the charging power or current drawn from a specific EVSE or the entire Charging Station over a period of time. The CSMS sends a SetChargingProfileRequest to the Charging Station to influence the power or current drawn by EVs. The CSMS calculates a ChargingSchedule to stay within certain limits, which MAY be imposed by any external system.	
Purpose	To verify if the Charging station is able to save a chargingProfile persistent over reboot as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SetChargingProfileResponse**	**1.** The OCTT sends a **SetChargingProfileRequest** with **evseId** *<Configured evseId>* AND **chargingProfile.id** *<Configured chargingProfileId>* AND **chargingProfile.chargingProfilePurpose** *TxDefaultProfile* **chargingProfile.chargingSchedule.duration** *<Configured duration>* **chargingProfile.chargingSchedule.chargingRateUnit** *<Configured chargingRateUnit>* **chargingProfile.chargingSchedule.chargingSchedulePeriod.startPeriod** *0* **chargingProfile.chargingSchedule.chargingSchedulePeriod.limit** *if unit is A then 6(A) else 6000(W)* **chargingProfile.chargingSchedule.chargingSchedulePeriod.numberPhases** *<Configured numberPhases>*
	3. Execute **Reusable State** *Booted*	
	5. The Charging Station responds with a **GetChargingProfilesResponse**	**4.** The OCTT sends a **GetChargingProfilesRequest** with **chargingProfile.chargingProfileId** *<Configured chargingProfileId>*
	6. The Charging Station sends a **ReportChargingProfilesRequest**	**7.** The OCTT responds with a **ReportChargingProfilesResponse**
	Note(s): *- If **tbc** is True at Step 6 then step 6 and 7 will be repeated*	

Test case name	**Set Charging Profile - Persistent over reboot**
Tool validations	* Step 2: Message **SetChargingProfileResponse** - **status** *Accepted* * Step 5: Message **GetChargingProfilesResponse** - **status** *Accepted* * Step 6: Message **ReportChargingProfilesRequest** - **requestId** *Same Id as in the GetChargingProfilesRequest in step 4* - **EVSEId** *<Configured EVSEId>* - **chargingProfile** *<Configured chargingProfile>*
	Post scenario validations: - The same profile is reported as send in step 1

Table 266. Test Case Id: TC_K_14_CS

Test case name	**Set Charging Profile - Unexisting EVSEid**	
Test case Id	TC_K_14_CS	
Use case Id(s)	K01	
Requirement(s)	K01.FR.28	
System under test	Charging Station	
Description	To enable the CSMS to influence the charging power or current drawn from a specific EVSE or the entire Charging Station over a period of time. The CSMS sends a SetChargingProfileRequest to the Charging Station to influence the power or current drawn by EVs. The CSMS calculates a ChargingSchedule to stay within certain limits, which MAY be imposed by any external system.	
Purpose	To verify if the Charging station is able to reject a chargingProfile when the provided EVSEid is unknown as described at the OCPP specification.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SetChargingProfileResponse**	**1.** The OCTT sends a **SetChargingProfileRequest** with **evseId** *<EVSECount + 1>*
Tool validations	* Step 2: Message **SetChargingProfileResponse** - **status** *Rejected*	
	Post scenario validations: - N/a	

Table 267. Test Case Id: TC_K_15_CS

Test case name	**Set Charging Profile - Not Supported**	
Test case Id	TC_K_15_CS	
Use case Id(s)	K01	
Requirement(s)	K01.FR.29	
System under test	Charging Station	
Description	To enable the CSMS to influence the charging power or current drawn from a specific EVSE or the entire Charging Station over a period of time. The CSMS sends a SetChargingProfileRequest to the Charging Station to influence the power or current drawn by EVs. The CSMS calculates a ChargingSchedule to stay within certain limits, which MAY be imposed by any external system.	
Purpose	To verify if the Charging station is able to raise a callerror when it does not support smart charging as described at the OCPP specification.	
Prerequisite(s)	Charging station does not support smart charging	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with RPC Framework: CALLERROR: NotSupported.	**1.** The OCTT sends a **SetChargingProfileRequest** with **chargingProfile.id** *<Configured chargingProfileId>*
Tool validations	- N/a	
	Post scenario validations: - N/a	

Table 268. Test Case Id: TC_K_16_CS

Test case name	**Set Charging Profile - Unknown transactionId**	
Test case Id	TC_K_16_CS	
Use case Id(s)	K01	
Requirement(s)	K01.FR.33	
System under test	Charging Station	
Description	To enable the CSMS to influence the charging power or current drawn from a specific EVSE or the entire Charging Station over a period of time. The CSMS sends a SetChargingProfileRequest to the Charging Station to influence the power or current drawn by EVs. The CSMS calculates a ChargingSchedule to stay within certain limits, which MAY be imposed by any external system.	
Purpose	To verify if the Charging station is able to reject a charge profile when an unknown transactionId is provided as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: **State is** *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SetChargingProfileResponse**	**1.** The OCTT sends a **SetChargingProfileRequest** with **evseId** *<Configured evseId>* AND **chargingProfile.id** *<Configured chargingProfileId>* AND **chargingProfile.chargingProfilePurpose** *TxProfile* AND **chargingProfile.transactionId** *UNKNOWN-TRANSACTION-ID*
Tool validations	* Step 2: Message **SetChargingProfileResponse** - **status** *Rejected*	
	Post scenario validations: - N/a	

Table 269. Test Case Id: TC_K_19_CS

Test case name	**Set Charging Profile - ChargingProfileKind is Recurring**	
Test case Id	TC_K_19_CS	
Use case Id(s)	K01	
Requirement(s)	K01.FR.40	
System under test	Charging Station	
Description	To enable the CSMS to influence the charging power or current drawn from a specific EVSE or the entire Charging Station over a period of time. The CSMS sends a SetChargingProfileRequest to the Charging Station to influence the power or current drawn by EVs. The CSMS calculates a ChargingSchedule to stay within certain limits, which MAY be imposed by any external system.	
Purpose	To verify if the Charging station is able to accept and successfully change to the Recurring ChargingProfileKind sent by the CSMS as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SetChargingProfileResponse**	**1.** The OCTT sends a **SetChargingProfileRequest** with **chargingProfile.chargingProfileKind** *Recurring* **chargingProfile.recurrencyKind** *<Configured RecurrencyKind>*
Tool validations	* Step 2: Message **SetChargingProfileResponse** - **status** *Accepted*	
	Post scenario validations: - N/a	

Table 270. Test Case Id: TC_K_21_CS

Test case name	**Set Charging Profile - ValidFrom**	
Test case Id	TC_K_21_CS	
Use case Id(s)	K01	
Requirement(s)	K01.FR.36	
System under test	Charging Station	
Description	The CSMS is able to impose charging limits by setting a charging profile that becomes valid after a certain date/time using the SetChargingProfileRequest message. It is only tested on EVSE #1, because mechanism is the same regardless of EVSE.	
Purpose	To verify if the Charging Station activates a set charging profile after the ValidFrom is reached.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SetChargingProfileResponse**	**1.** The OCTT sends a **SetChargingProfileRequest** with **chargingProfile.chargingProfilePurpose** is *TxDefaultProfile* **chargingProfile.chargingProfileKind** is *Relative* **evseId** *<configured evseId>* **chargingProfile.validFrom** *<current dateTime + 300 seconds>* **chargingProfile.validTo** is absent **chargingProfile.chargingSchedule[0].startSchedule** is absent **chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].numberPhases** *<Configured numberPhases>* **chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].startPeriod** *0* If *<Configured chargingRateUnit>* is *A*: **chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].limit** *6* If *<Configured chargingRateUnit>* is *W*: **chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].limit** *6000*
	4. The Charging Station responds with a **GetCompositeScheduleResponse**	**3.** The OCTT sends a **GetCompositeScheduleRequest** with **evseId** *<configured evseId>* **duration** is *400* **chargingRateUnit** *<Configured chargingRateUnit>*

Test case name	**Set Charging Profile - ValidFrom**
Tool validations	* Step 2: (Message: **SetChargingProfileResponse**) **status** is *Accepted* * Step 4: (Message: **GetCompositeScheduleResponse**) **status** *Accepted* **schedule.evseId** *<configured evseId>* **schedule.chargingRateUnit** *<Configured chargingRateUnit>* **schedule.duration** *400* **schedule.chargingSchedulePeriod[0].startPeriod** *0*, **schedule.chargingSchedulePeriod[1].startPeriod** *(300 - x)*, **schedule.chargingSchedulePeriod[1].limit** *6.0* *Note: If <Configured chargingRateUnit> is W, then the **limit** field will be multiplied by 1000.* *Note: The period of time between sending the second SetChargingProfileRequest and the **scheduleStart** from the GetCompositeScheduleResponse is called **x**:*
	Post scenario validations: N/a

Table 271. Test Case Id: TC_K_22_CS

Test case name	**Set Charging Profile - ValidTo**	
Test case Id	TC_K_22_CS	
Use case Id(s)	K01	
Requirement(s)	K01.FR.37	
System under test	Charging Station	
Description	The CSMS is able to impose charging limits by setting a charging profile using the SetChargingProfileRequest message.	
Purpose	To verify if the Charging Station deactivates a set charging profile after the ValidTo has passed.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SetChargingProfileResponse**	**1.** The OCTT sends a **SetChargingProfileRequest** with **chargingProfile.chargingProfilePurpose** is *TxDefaultProfile* **chargingProfile.chargingProfileKind** is *Absolute* **evseId** *0* **chargingProfile.validFrom** *<current dateTime - <Configured max time deviation> seconds>* **chargingProfile.validTo** *<current dateTime + 300 seconds>* **chargingProfile.chargingSchedule[0].startSchedule** *<current dateTime>* **chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].numberPhases** *<Configured numberPhases>* **chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].startPeriod** *0* If *<Configured chargingRateUnit>* is *A*: **chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].limit** *6* If *<Configured chargingRateUnit>* is *W*: **chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].limit** *6000*
	4. The Charging Station responds with a **GetCompositeScheduleResponse**	**3.** The OCTT sends a **GetCompositeScheduleRequest** with **evseId** *i* **duration** is *400* **chargingRateUnit** *<Configured chargingRateUnit>*
	Note(s): *- Steps 3 and 4 are repeated for i= 0, 1, …, nr. of configured EVSEs*	

Test case name	**Set Charging Profile - ValidTo**
Tool validations	* Step 2: (Message: **SetChargingProfileResponse**) **status** is *Accepted* * Step 4: (Message: **GetCompositeScheduleResponse**) **status** *Accepted* **evseId** *<requested evseId>* **chargingRateUnit** *<Configured chargingRateUnit>* **ChargingSchedule**: **duration** *400* **chargingRateUnit** *<Configured chargingRateUnit>* *Note: If <Configured chargingRateUnit> is W, then the **limit** field will be multiplied by 1000.* *Note: The period of time between sending the second SetChargingProfileRequest and the **scheduleStart** from the GetCompositeScheduleResponse is called **x**:* **startPeriod** *0*, **limit** *6 (for evse 0 the limit is multiplied by the nr. of EVSE)*+ **startPeriod** *(300 - x)*, **limit** *<Local limit of Charging Station (This is not validated)>*
	Post scenario validations: N/a

Table 272. Test Case Id: TC_K_23_CS

Test case name	**Set Charging Profile - StartSchedule**	
Test case Id	TC_K_23_CS	
Use case Id(s)	K01	
Requirement(s)	K01.FR.30	
System under test	Charging Station	
Description	The CSMS is able to impose charging limits by setting a charging profile using the SetChargingProfileRequest message.	
Purpose	To verify if the Charging Station activates a set charging profile after the StartSchedule has passed.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SetChargingProfileResponse**	**1.** The OCTT sends a **SetChargingProfileRequest** with **chargingProfile.chargingProfilePurpose** is *TxDefaultProfile* **chargingProfile.chargingProfileKind** is *Absolute* **evseId** *<configured evseId>* **chargingProfile.validFrom** *<current dateTime - <Configured max time deviation> + 50 seconds>* **chargingProfile.validTo** *<current dateTime + <Configured max time deviation> + 400 seconds>* **chargingProfile.chargingSchedule[0].startSchedule** *<current dateTime - <Configured max time deviation> + 60 seconds>* **chargingProfile.chargingSchedule[0].chargingSchedulePeriod[].numberPhases** *<Configured numberPhases>* **chargingProfile.chargingSchedule[0].chargingSchedulePeriod[].startPeriod** *0* If *<Configured chargingRateUnit>* is *A*: **chargingProfile.chargingSchedule[0].chargingSchedulePeriod[].limit** *6* If *<Configured chargingRateUnit>* is *W*: **chargingProfile.chargingSchedule[0].chargingSchedulePeriod[].limit** *6000*
	4. The Charging Station responds with a **GetCompositeScheduleResponse**	**3.** The OCTT sends a **GetCompositeScheduleRequest** with **evseId** *<Configured evseId>* **duration** is *300* **chargingRateUnit** *<Configured chargingRateUnit>*

Test case name	**Set Charging Profile - StartSchedule**
Tool validations	* Step 2: (Message: **SetChargingProfileResponse**) **status** is *Accepted* * Step 4: (Message: **GetCompositeScheduleResponse**) **status** *Accepted* **evseId** *<Configured evseId>* **ChargingSchedule**: **duration** *300* **chargingRateUnit** *<Configured chargingRateUnit>* *Note: If <Configured chargingRateUnit> is W, then the **limit** field will be multiplied by 1000.* *Note: The period of time between sending the second SetChargingProfileRequest and the **scheduleStart** from the GetCompositeScheduleResponse is called **x**:* **startPeriod** *0*, **limit** *<Local limit of Charging Station (This is not validated)>* **startPeriod** *(60 - x)*, **limit** *6*
	Post scenario validations: N/a

Table 273. Test Case Id: TC_K_24_CS

Test case name	**Clear Charging Profile - With stackLevel/purpose combination for multiple profiles**	
Test case Id	TC_K_24_CS	
Use case Id(s)	K10	
Requirement(s)	K10.FR.04	
System under test	Charging Station	
Description	If the CSMS wishes to clear some or all of the charging profiles that were previously sent to the Charging Station, then the CSMS sends a ClearChargingProfileRequest to the Charging Station.	
Purpose	To verify if the Charging station is able to accept the request and clear charging profiles sent with a stackLevel/purpose combination by the CSMS as described at the OCPP specification.	
Prerequisite(s)	Charging Station needs to have 2 or more EVSE.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: A chargingprofile with *<Configured chargingProfilePurpose>* AND *<Configured stackLevel>* is configured for evseId 1. A chargingprofile with *<Configured chargingProfilePurpose>* AND *<Configured stackLevel>* is configured for evseId 2.	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **ClearChargingProfileResponse**	**1.** The OCTT sends a **ClearChargingProfileRequest** with **chargingProfileCriteria.chargingProfilePurpose** *<Configured chargingProfilePurpose>* AND **chargingProfileCriteria.stackLevel** *<Configured stackLevel>*
	4. The Charging Station responds with a **GetChargingProfilesResponse**	**3.** The OCTT sends a **GetChargingProfilesRequest** with **chargingProfile.chargingProfilePurpose** *<Configured chargingProfilePurpose>* AND **chargingProfile.stackLevel** *<Configured stackLevel>*
Tool validations	* Step 2: Message **ClearChargingProfileResponse** - **status** *Accepted* * Step 4: Message **GetChargingProfilesResponse** - **status** *NoProfiles*	
	Post scenario validations: - N/a	

Table 274. Test Case Id: TC_K_28_CS

Test case name	**Set Charging Profile - TxDefaultProfile with transaction ongoing**	
Test case Id	TC_K_28_CS	
Use case Id(s)	K01	
Requirement(s)	K01.FR.32	
System under test	Charging Station	
Description	The CSMS sets a default schedule for a currently ongoing transaction.	
Purpose	To verify if the CSMS and Charging Station are able to exchange messages to set a default schedule for a currently ongoing transaction.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** **SmartChargingCtrlr.LimitChangeSignificance** is *1.0*	
	Memory State: N/a	
	Reusable State(s): **State is** *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SetChargingProfileResponse**	**1.** The OCTT sends a **SetChargingProfileRequest** with **chargingProfile.chargingProfilePurpose** is *TxDefaultProfile* **chargingProfile.chargingProfileKind** is *Absolute* **chargingProfile.chargingSchedule[0].duration** is *300* **evseId** *<Configured evseId>* **chargingProfile.validFrom** *<current dateTime - <Configured max time deviation> seconds>* **chargingProfile.validTo** *<current dateTime + <Configured max time deviation> + 300 seconds>* **chargingProfile.chargingSchedule[0].startSchedule** *<current dateTime - <Configured max time deviation> seconds>* **chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].numberPhases** *<Configured numberPhases>* **chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].startPeriod** *0* If *<Configured chargingRateUnit>* is *A*: **chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].limit** *6* If *<Configured chargingRateUnit>* is *W*: **chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].limit** *6000*
	4. The Charging Station responds with a **GetCompositeScheduleResponse**	**3.** The OCTT sends a **GetCompositeScheduleRequest** with **evseId** *<Configured evseId>* **duration** is *300* **chargingRateUnit** *<Configured chargingRateUnit>*

Test case name	**Set Charging Profile - TxDefaultProfile with transaction ongoing**
Tool validations	* Step 2: (Message: **SetChargingProfileResponse**) **status** is *Accepted* * Step 4: (Message: **GetCompositeScheduleResponse**) **status** *Accepted* **evseId** *<Configured evseId>* **ChargingSchedule**: **duration** *300* **chargingRateUnit** *<Configured chargingRateUnit>* *Note: If <Configured chargingRateUnit> is W, then the* ***limit*** *field will be multiplied by 1000.* *Note: The period of time between sending the second SetChargingProfileRequest and the* ***scheduleStart*** *from the GetCompositeScheduleResponse is called* ***x****:* **startPeriod** *0*, **limit** *<6/6000>*
	Post scenario validations: N/a

Table 275. Test Case Id: TC_K_29_CS

Test case name	**Get Charging Profile - EvseId 0**	
Test case Id	TC_K_29_CS	
Use case Id(s)	K09	
Requirement(s)	K09.FR.02, K09.FR.05	
System under test	Charging Station	
Description	With the GetChargingProfilesRequest message the CSMS can ask a Charging Station to report all, or a subset of all the install Charging Profiles from the different possible sources. This can be used for some automatic smart charging control system, or for debug purposes by a CSO.	
Purpose	To verify if the Charging station is able to successfully report the charging profile(s) requested as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: Charging station has a charging profile with *<Generated Id1>* AND **chargingProfilePurpose** *ChargingStationMaxProfile* configured on the charging statation. Charging station has a second charge profile with *<Generated Id2>* AND **chargingProfilePurpose** *TxDefaultProfile* configured on *<Configured evseId>*.	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **GetChargingProfilesResponse**	**1.** The OCTT sends a **GetChargingProfilesRequest** with **evseId** *0*
	3. The Charging Station sends a **ReportChargingProfilesRequest**	**4.** The OCTT responds with a **ReportChargingProfilesResponse**
	Note(s): - *If **tbc** is True at Step 3 then step 3 and 4 will be repeated*	
Tool validations	* Step 2: Message **GetChargingProfilesResponse** - **status** *Accepted* * Step 3: Message **ReportChargingProfilesRequest** - **requestId** *<Generated requestId>* - **chargingProfile** *<Generated ChargingProfile1>* with **chargingProfilePurpose** *ChargingStationMaxProfile*	
	Post scenario validations: - All report message have been received	

Table 276. Test Case Id: TC_K_30_CS

Test case name	**Get Charging Profile - EvseId > 0**	
Test case Id	TC_K_30_CS	
Use case Id(s)	K09	
Requirement(s)	K09.FR.02, K09.FR.04	
System under test	Charging Station	
Description	With the GetChargingProfilesRequest message the CSMS can ask a Charging Station to report all, or a subset of all the install Charging Profiles from the different possible sources. This can be used for some automatic smart charging control system, or for debug purposes by a CSO.	
Purpose	To verify if the Charging station is able to successfully report the charging profile(s) requested for a specific EVSE as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: Charging station has a charging profile with *<Generated Id1>* AND *ChargingStationMaxProfile* configured on the charging station. Charging station has a second charge profile with *<Generated Id2>* AND *TxDefaultProfile* configured on *<Configured evseId>*.	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **GetChargingProfilesResponse**	**1.** The OCTT sends a **GetChargingProfilesRequest** with **evseId** *<Configured evseId>*
	3. The Charging Station sends a **ReportChargingProfilesRequest**	**4.** The OCTT responds with a **ReportChargingProfilesResponse**
	Note(s): *- If **tbc** is True at Step 3 then step 3 and 4 will be repeated*	
Tool validations	* Step 2: Message **GetChargingProfilesResponse** - **status** *Accepted* * Step 3: Message **ReportChargingProfilesRequest** - **requestId** *<Generated requestId>* - **chargingProfile** *<Generated ChargingProfile>*	
	Post scenario validations: - All report message have been received	

Table 277. Test Case Id: TC_K_31_CS

Test case name	**Get Charging Profile - No EvseId**	
Test case Id	TC_K_31_CS	
Use case Id(s)	K09	
Requirement(s)	K09.FR.02, K09.FR.06	
System under test	Charging Station	
Description	With the GetChargingProfilesRequest message the CSMS can ask a Charging Station to report all, or a subset of all the install Charging Profiles from the different possible sources. This can be used for some automatic smart charging control system, or for debug purposes by a CSO.	
Purpose	To verify if the Charging station is able to successfully report all installed charging profiles requested as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: Charging station has a charging profile with *<Generated Id1>* AND *ChargingStationMaxProfile* configured on the charging station. Charging station has a second charge profile with *<Generated Id2>* AND *TxDefaultProfile* configured on **EVSEId** *<Configured evseId>*.	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **GetChargingProfilesResponse**	**1.** The OCTT sends a **GetChargingProfilesRequest** with: **requestId** *Generated requestId*
	3. The Charging Station sends a **ReportChargingProfilesRequest**	**4.** The OCTT responds with a **ReportChargingProfilesResponse**
	Note(s): *- If **tbc** is True at Step 3 then step 3 and 4 will be repeated*	
Tool validations	* Step 2: Message **GetChargingProfilesResponse** - **status** *Accepted* * Step 3: Message **ReportChargingProfilesRequest** - **requestId** *<Generated requestId>* - **chargingProfiles** *<Configured ChargingProfiles>*	
	Post scenario validations: - All report message have been received	

Table 278. Test Case Id: TC_K_32_CS

Test case name	**Get Charging Profile - chargingProfileId**	
Test case Id	TC_K_32_CS	
Use case Id(s)	K09	
Requirement(s)	K09.FR.01, K09.FR.02	
System under test	Charging Station	
Description	With the GetChargingProfilesRequest message the CSMS can ask a Charging Station to report all, or a subset of all the install Charging Profiles from the different possible sources. This can be used for some automatic smart charging control system, or for debug purposes by a CSO.	
Purpose	To verify if the Charging station is able to successfully report a specific charging profile requested as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: Charging station has a charging profile with *<Generated Id1>* AND *ChargingStationMaxProfile* configured on the charging station. Charging station has a second charge profile with *<Generated Id2>* AND *TxDefaultProfile* configured on **EVSEId** *<Configured evseId>*.	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **GetChargingProfilesResponse**	**1.** The OCTT sends a **GetChargingProfilesRequest** with **chargingProfileId** *<Generated Id1>*
	3. The Charging Station sends a **ReportChargingProfilesRequest**	**4.** The OCTT responds with a **ReportChargingProfilesResponse**
	Note(s): *- If **tbc** is True at Step 3 then step 3 and 4 will be repeated*	
Tool validations	* Step 2: Message **GetChargingProfilesResponse** - **status** *Accepted* * Step 3: Message **ReportChargingProfilesRequest** - **requestId** *Generated Id1* - **chargingProfile** *<Configured chargingProfile>*	
	Post scenario validations: - All report message have been received	

Table 279. Test Case Id: TC_K_33_CS

Test case name	**Get Charging Profile - EvseId > 0 + stackLevel**	
Test case Id	TC_K_33_CS	
Use case Id(s)	K09	
Requirement(s)	K09.FR.02, K09.FR.04	
System under test	Charging Station	
Description	With the GetChargingProfilesRequest message the CSMS can ask a Charging Station to report all, or a subset of all the install Charging Profiles from the different possible sources. This can be used for some automatic smart charging control system, or for debug purposes by a CSO.	
Purpose	To verify if the Charging station is able to successfully report a charging profile with specific stackLevel requested for a specific EVSE as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: Charging station has a charging profile with *<Generated Id1>* AND *ChargingStationMaxProfile* AND *<Configured stackLevel>* configured on the station. Charging station has a second charge profile with *<Generated Id2>* AND *TxDefaultProfile* AND *<Configured stackLevel2>* configured on *<Configured evseId>*.	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **GetChargingProfilesResponse**	**1.** The OCTT sends a **GetChargingProfilesRequest** with **evseId** *<Configured evseId>* AND **chargingProfile.stackLevel** *<Configured stackLevel>*
	3. The Charging Station sends a **ReportChargingProfilesRequest**	**4.** The OCTT responds with a **ReportChargingProfilesResponse**
	Note(s): *- If **tbc** is True at Step 3 then step 3 and 4 will be repeated*	
Tool validations	* Step 2: Message **GetChargingProfilesResponse** - **status** *Accepted* * Step 3: Message **ReportChargingProfilesRequest** - **requestId** *Generated Id1* - **chargingProfile** *<Configured ChargingProfile>*	
	Post scenario validations: - All report message have been received	

Table 280. Test Case Id: TC_K_34_CS

Test case name	**Get Charging Profile - EvseId > 0 + chargingLimitSource**	
Test case Id	TC_K_34_CS	
Use case Id(s)	K09	
Requirement(s)	K09.FR.02, K09.FR.04	
System under test	Charging Station	
Description	With the GetChargingProfilesRequest message the CSMS can ask a Charging Station to report all, or a subset of all the install Charging Profiles from the different possible sources. This can be used for some automatic smart charging control system, or for debug purposes by a CSO.	
Purpose	To verify if the Charging station is able to successfully report a charging profile with specific chargingLimitSource requested for a specific EVSE as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: *<Configured chargingLimitSource>* should be *CSO* AND *<Configured chargingLimitSource2>* should have no existing profiles AND Charging station has a charging profile with: - **id** *<Generated Id1>* - **chargingProfilePurpose** *TxDefaultProfile* - **stackLevel** *<Configured StackLevel + 1>*	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **GetChargingProfilesResponse**	**1.** The OCTT sends a **GetChargingProfilesRequest** with **evseId** *<Configured evseId>* AND **chargingProfile.chargingLimitSource** *<Configured chargingLimitSource>*
	3. The Charging Station sends a **ReportChargingProfilesRequest**	**4.** The OCTT responds with a **ReportChargingProfilesResponse**
	Note(s): *- If **tbc** is True at Step 3 then step 3 and 4 will be repeated*	
	6. The Charging Station responds with a **GetChargingProfilesResponse**	**5.** The OCTT sends a **GetChargingProfilesRequest** with **evseId** *<Configured evseId>* AND **chargingProfile.chargingLimitSource** *<Configured chargingLimitSource2>*
Tool validations	* Step 2: Message **GetChargingProfilesResponse** - **status** *Accepted* * Step 3: Message **ReportChargingProfilesRequest** - **requestId** *Generated Id1* - **chargingProfile** *<ChargingProfile>* * Step 6: Message **GetChargingProfilesResponse** - **status** *NoProfiles*	
	Post scenario validations: - All report message have been received	

Table 281. Test Case Id: TC_K_35_CS

Test case name	**Get Charging Profile - EvseId > 0 + chargingProfilePurpose**	
Test case Id	TC_K_35_CS	
Use case Id(s)	K09	
Requirement(s)	K09.FR.02, K09.FR.04	
System under test	Charging Station	
Description	With the GetChargingProfilesRequest message the CSMS can ask a Charging Station to report all, or a subset of all the install Charging Profiles from the different possible sources. This can be used for some automatic smart charging control system, or for debug purposes by a CSO.	
Purpose	To verify if the Charging station is able to successfully report a charging profile with specific chargingProfilePurpose requested for a specific EVSE as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: Charging station has a charge profile with *<Generated Id1>* AND *ChargingStationMaxProfile* configured on the charging station. Charging station has a second charge profile with *<Generated Id2>* AND *TxDefaultProfile* configured on *<Configured evseId>*.	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **GetChargingProfilesResponse**	**1.** The OCTT sends a **GetChargingProfilesRequest** with **evseId** *<Configured evseId>* AND **chargingProfile.chargingProfilePurpose** *<Configured chargingProfilePurpose>*
	3. The Charging Station sends a **ReportChargingProfilesRequest**	**4.** The OCTT responds with a **ReportChargingProfilesResponse**
	Note(s): - *If **tbc** is True at Step 3 then step 3 and 4 will be repeated*	
Tool validations	* Step 2: Message **GetChargingProfilesResponse** - **status** *Accepted* * Step 3: Message **ReportChargingProfilesRequest** - **requestId** *Generated Id1* - **ChargingProfile** *<Configured ChargingProfile>*	
	Post scenario validations: - All report message have been received	

Table 282. Test Case Id: TC_K_36_CS

Test case name	**Get Charging Profile - EvseId > 0 + chargingProfilePurpose + stackLevel**	
Test case Id	TC_K_36_CS	
Use case Id(s)	K09	
Requirement(s)	K09.FR.02, K09.FR.04	
System under test	Charging Station	
Description	With the GetChargingProfilesRequest message the CSMS can ask a Charging Station to report all, or a subset of all the install Charging Profiles from the different possible sources. This can be used for some automatic smart charging control system, or for debug purposes by a CSO.	
Purpose	To verify if the Charging station is able to successfully report a charging profile with specific chargingProfilePurpose and stackLevel requested for a specific EVSE as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: Charging station has a charge profile with *<Generated Id1>* AND *ChargingStationMaxProfile* AND *<Configured stackLevel>* configured on the charging station. Charging station has a second charge profile with *<Generated Id2>* AND *TxDefaultProfile* AND *<Configured stackLevel>* configured on *<Configured evseId>*.	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **GetChargingProfilesResponse**	**1.** The OCTT sends a **GetChargingProfilesRequest** with **evseId** *<Configured evseId>* AND **chargingProfile.chargingProfilePurpose** *<TxDefaultProfile>* AND **chargingProfile.stackLevel** *<Configured stackLevel>*
	3. The Charging Station sends a **ReportChargingProfilesRequest**	**4.** The OCTT responds with a **ReportChargingProfilesResponse**
	Note(s): *- If **tbc** is True at Step 3 then step 3 and 4 will be repeated*	
Tool validations	* Step 2: Message **GetChargingProfilesResponse** - **status** *Accepted* * Step 3: Message **ReportChargingProfilesRequest** - **requestId** *Generated Id1* - **ChargingProfile** *<Configured ChargingProfile>*	
	Post scenario validations: - All report message have been received	

Table 283. Test Case Id: TC_K_60_CS

Test case name	**Set Charging Profile - TxProfile with ongoing transaction on the specified EVSE**	
Test case Id	TC_K_60_CS	
Use case Id(s)	K01	
Requirement(s)	K01.FR.04, K01.FR.07, K01.FR.15	
System under test	Charging Station	
Description	The CSMS sets a TxProfile on a specific EVSE for a currently ongoing transaction.	
Purpose	To verify if the Charging Station is able to exchange messages to set a TxProfile on a specific EVSE for a currently ongoing transaction.	
Prerequisite(s)	The Charging Station must support the GetChargingProfiles feature.	
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 **SetChargingProfileResponse**	**1.** The OCTT sends a **SetChargingProfileRequest** with **chargingProfile.chargingProfilePurpose** is *TxProfile* **chargingProfile.transactionId** is *<transactionId returned by Charging Station in before>* **chargingProfile.chargingProfileKind** is *Relative* **evseId** *<Configured evseId>* **chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].numberPhases** *<Configured numberPhases>* **chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].startPeriod** *0* If *<Configured chargingRateUnit>* is *A*: **chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].limit** *6* If *<Configured chargingRateUnit>* is *W*: **chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].limit** *6000*
	4. The Charging Station responds with a **GetChargingProfilesResponse**	**3.** The OCTT sends a **GetChargingProfilesRequest** with **chargingProfile.chargingProfileId** *<Used chargingProfileId at step 1>*
	5. The Charging Station sends a **ReportChargingProfilesRequest**	**6.** The OCTT responds with a **ReportChargingProfilesResponse**
Tool validations	* Step 2: (Message: **SetChargingProfileResponse**) **status** is *Accepted* * Step 4: (Message: **GetChargingProfilesResponse**) **status** is *Accepted* * Step 5: (Message: **ReportChargingProfilesRequest**) **chargingProfile** *<The Charging Profile set at step 1>*	
	Post scenario validations: N/a	

Table 284. Test Case Id: TC_K_37_CS

Test case name	**Remote start transaction with charging profile - Success**	
Test case Id	TC_K_37_CS	
Use case Id(s)	K05,F01	
Requirement(s)	K05.FR.03, E01.FR.02,F01.FR.10,F01.FR.13	
System under test	Charging Station	
Description	The CSMS sets a TxProfile on a specific EVSE inside a RequestStartTransactionRequest message.	
Purpose	To verify if the Charging Station is able to set a TxProfile on a specific EVSE when receiving one in a RequestStartTransactionRequest message.	
Prerequisite(s)	The Charging Station must support the GetChargingProfiles feature.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): **State is** *EVConnectedPreSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **RequestStartTransactionResponse**	**1.** The OCTT sends a **RequestStartTransactionRequest** with **idToken.idToken** *<Configured valid_idtoken_idtoken>* **idToken.type** *<Configured valid_idtoken_type>* **evseId** *<Configured evseId>* **chargingProfile.chargingProfilePurpose** is *TxProfile* **chargingProfile.transactionId** is omitted. **chargingProfile.chargingProfileKind** is *Relative* **chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].numberPhases** *<Configured numberPhases>* **chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].startPeriod** *0* If *<Configured chargingRateUnit>* is *A*: **chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].limit** *6* If *<Configured chargingRateUnit>* is *W*: **chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].limit** *6000*
	3. The Charging Station sends an **AuthorizeRequest** Note(s): *- This step needs to be executed when **AuthCtrlr.AuthorizeRemoteStart** is true, unless (**AuthEnabled** is implemented with mutability ReadOnly AND the value is set to false) OR the **idToken** is cached.* *In case the **idToken** is used for a reservation, sending the **AuthorizeRequest** message is optional.*	**4.** The OCTT responds with an **AuthorizeResponse** with **idTokenInfo.status** *Accepted*

Test case name	**Remote start transaction with charging profile - Success**	
	5. The Charging Station sends a **TransactionEventRequest** Note(s): *- This step needs to be executed when* ***TxStartPoint*** *contains Authorized OR the transaction already started. So in the case* ***TxStartPoint*** *contains ParkingBayOccupancy or (EVConnected, in the case this testcase was initiated from state EVConnectedPreSession.)*	**6.** The OCTT responds with a **TransactionEventResponse** Note(s): *- The first TransactionEventRequest sent after authorization contains the idToken field. The TransactionEventResponse of this request message contains* ***idTokenInfo*** *with* ***status*** *Accepted*
	7. Execute **Reusable State** *EnergyTransferStarted*	
	9. The Charging Station responds with a **GetChargingProfilesResponse**	**8.** The OCTT sends a **GetChargingProfilesRequest** with **chargingProfile.chargingProfileId** *<Used chargingProfileId at step 1>*
	10. The Charging Station sends a **ReportChargingProfilesRequest**	**11.** The OCTT responds with a **ReportChargingProfilesResponse**
Tool validations	* Step 2: Message: **RequestStartTransactionResponse** - **status** must be *Accepted* If the transaction has already been started, so if TxStartPoint contains *ParkingBayOccupancy* OR (<Configured TxStartPoint> contains *EVConnected* AND **State** pre reusable state execution was *EVConnectedPreSession*) then - **transactionId** must be *<Provided transactionId in first TransactionEventRequest>* * Step 3: Message: **AuthorizeRequest** - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>* * Step 5: Message: **TransactionEventRequest** - **triggerReason** must be *RemoteStart* - **transactionInfo.remoteStartId** must be present. * Step 9: (Message: **GetChargingProfilesResponse**) **status** is *Accepted* * Step 10: (Message: **ReportChargingProfilesRequest**) **chargingProfile** *<The Charging Profile set at step 1>*	
	Post scenario validations: N/a	

Table 285. Test Case Id: TC_K_38_CS

Test case name	**Remote start transaction with charging profile - Ignore chargingProfile**	
Test case Id	TC_K_38_CS	
Use case Id(s)	F01	
Requirement(s)	F01.FR.12,F01.FR.13	
System under test	Charging Station	
Description	The CSMS sets a TxProfile on a specific EVSE inside a RequestStartTransactionRequest message.	
Purpose	To verify if the Charging Station is able to ignore a TxProfile on a specific EVSE when receiving one in a RequestStartTransactionRequest message, when it does not support Smart Charging.	
Prerequisite(s)	The Charging Station does NOT support Smart Charging.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): **State is** *EVConnectedPreSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **RequestStartTransactionResponse**	**1.** The OCTT sends a **RequestStartTransactionRequest** with **idToken.idToken** *<Configured valid_idtoken_idtoken>* **idToken.type** *<Configured valid_idtoken_type>* **evseId** *<Configured evseId>* **chargingProfile.chargingProfilePurpose** is *TxProfile* **chargingProfile.transactionId** is omitted. **chargingProfile.chargingProfileKind** is *Relative* **chargingProfile.chargingSchedule.chargingSchedulePeriod[0].numberPhases** *<Configured numberPhases>* **chargingProfile.chargingSchedule.chargingSchedulePeriod[0].startPeriod** *0* If *<Configured chargingRateUnit>* is *A*: **chargingProfile.chargingSchedule.chargingSchedulePeriod[0].limit** *6* If *<Configured chargingRateUnit>* is *W*: **chargingProfile.chargingSchedule.chargingSchedulePeriod[0].limit** *6000*
	3. The Charging Station sends an **AuthorizeRequest** Note(s): *- This step needs to be executed when **AuthCtrlr.AuthorizeRemoteStart** is true, unless (**AuthEnabled** is implemented with mutability ReadOnly AND the value is set to false) OR the **idToken** is cached.* *In case the **idToken** is used for a reservation, sending the **AuthorizeRequest** message is optional.*	**4.** The OCTT responds with an **AuthorizeResponse** with **idTokenInfo.status** *Accepted*

Test case name	**Remote start transaction with charging profile - Ignore chargingProfile**	
	5. The Charging Station sends a **TransactionEventRequest** Note(s): *- This step needs to be executed when **TxStartPoint** contains Authorized OR the transaction already started. So in the case **TxStartPoint** contains ParkingBayOccupancy or (EVConnected, in the case this testcase was initiated from state EVConnectedPreSession.)*	**6.** The OCTT responds with a **TransactionEventResponse** Note(s): *- The first TransactionEventRequest sent after authorization contains the idToken field. The TransactionEventResponse of this request message contains **idTokenInfo** with **status** Accepted*
	7. Execute **Reusable State** *EnergyTransferStarted*	
Tool validations	* Step 2: Message: **RequestStartTransactionResponse** - **status** must be *Accepted* If the transaction has already been started, so if TxStartPoint contains *ParkingBayOccupancy* OR (TxStartPoint contains *EVConnected* AND **State** pre reusable state execution was *EVConnectedPreSession*) then - **transactionId** must be *<Provided transactionId in first TransactionEventRequest>* * Step 3: Message: **AuthorizeRequest** - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>* * Step 5: Message: **TransactionEventRequest** - **triggerReason** must be *RemoteStart* - **transactionInfo.remoteStartId** must be present.	
	Post scenario validations: N/a	

Table 286. Test Case Id: TC_K_39_CS

Test case name	**Get Composite Schedule - No ChargingProfile installed on Charging Station**	
Test case Id	TC_K_39_CS	
Use case Id(s)	K08	
Requirement(s)	K08.FR.02, K08.FR.03,K08.FR.06	
System under test	Charging Station	
Description	The CSMS requests a composite schedule which is a combination of local limits and the prevailing Charging Profiles of the different chargingProfilePurposes and stack levels.	
Purpose	To verify if the Charging Station is able to calculate a correct composite schedule and provide this to the CSMS on request.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **GetCompositeScheduleResponse**	**1.** The OCTT sends a **GetCompositeScheduleRequest** with **evseId** *0* **duration** is *300* **chargingRateUnit** *<Configured chargingRateUnit>*
Tool validations	* Step 2: (Message: **GetCompositeScheduleResponse**) **status** *Accepted* **evseId** *0* **duration** is *300* **chargingRateUnit** *<Configured chargingRateUnit>* **startPeriod** *0*	
	Post scenario validations: N/a	

Table 287. Test Case Id: TC_K_40_CS

Test case name	**Get Composite Schedule - Stacking ChargingProfiles**
Test case Id	TC_K_40_CS
Use case Id(s)	K08
Requirement(s)	K08.FR.02,K08.FR.06
System under test	Charging Station
Description	The CSMS requests a composite schedule which is a combination of local limits and the prevailing Charging Profiles of the different chargingProfilePurposes and stack levels.
Purpose	To verify if the Charging Station is able to calculate a correct composite schedule and provide this to the CSMS on request.
Prerequisite(s)	- **ChargingProfileEntries.maxLimit** must be > 1 - The configuration variable **ChargingProfileMaxStackLevel** must be > 0 - The configuration variable **PeriodsPerSchedule** must be > 2
Before (Preparations)	**Configuration State:** N/a
	Memory State: *SetChargingProfile* with ChargingProfile 1: **chargingProfilePurpose** is *TxDefaultProfile* **chargingProfileKind** should be *Absolute* **stackLevel** should be *0* **evseId** *<Configured evseId>* **validFrom** *<current dateTime - <Configured max time deviation> seconds>* **validTo** *<current dateTime + <Configured max time deviation> + 401 seconds>* **startSchedule** *<current dateTime - <Configured max time deviation> seconds>* **numberPhases** *<Configured numberPhases>* **ChargingSchedule**: **duration** *400* + <Configured max time deviation> **chargingRateUnit** *<Configured chargingRateUnit>* *Note: If <Configured chargingRateUnit> is W, then the **limit** field will be multiplied by 1000.* **startPeriod** *0*, **limit** *6* **startPeriod** *100*, **limit** *8* **startPeriod** *200*, **limit** *10* ChargingProfile 2: **chargingProfilePurpose** is *TxDefaultProfile* **chargingProfileKind** should be *Absolute* **stackLevel** should be *1* **evseId** *<Configured evseId>* **validFrom** *<current dateTime - <Configured max time deviation> seconds>* **validTo** *<current dateTime + <Configured max time deviation> + 401 seconds>* **startSchedule** *<current dateTime - <Configured max time deviation> seconds>* **numberPhases** *<Configured numberPhases>* **ChargingSchedule**: **duration** *150* + <Configured max time deviation> **chargingRateUnit** *<Configured chargingRateUnit>* *Note: If <Configured chargingRateUnit> is W, then the **limit** field will be multiplied by 1000.* **startPeriod** *0*, **limit** *7* **startPeriod** *100*, **limit** *9*
	Reusable State(s): N/a

Test case name	**Get Composite Schedule - Stacking ChargingProfiles**	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **GetCompositeScheduleResponse**	**1.** The OCTT sends a **GetCompositeScheduleRequest** with **evseId** *<Configured evseId>* **duration** is *350* **chargingRateUnit** *<Configured chargingRateUnit>*
Tool validations	* Step 2: (Message: **GetCompositeScheduleResponse**) **status** *Accepted* **evseId** *<Configured evseId>* **ChargingSchedule**: **duration** *350* **chargingRateUnit** *<Configured chargingRateUnit>* *Note: If <Configured chargingRateUnit> is W, then the **limit** field will be multiplied by 1000.* *Note: The period of time between sending the second SetChargingProfileRequest and the **scheduleStart** from the GetCompositeScheduleResponse is called **x**:* ***startPeriod** _0*, **limit** *7* **startPeriod** *(100 - x)*, **limit** *9* **startPeriod** *(150 - x)*, **limit** *8* **startPeriod** *(200 - x)*, **limit** *10*	
	Post scenario validations: N/a	

Table 288. Test Case Id: TC_K_41_CS

Test case name	**Get Composite Schedule - Combining chargingProfilePurposes**	
Test case Id	TC_K_41_CS	
Use case Id(s)	K08	
Requirement(s)	K08.FR.02,K08.FR.04	
System under test	Charging Station	
Description	The CSMS requests a composite schedule which is a combination of local limits and the prevailing Charging Profiles of the different chargingProfilePurposes and stack levels.	
Purpose	To verify if the Charging Station is able to calculate a correct composite schedule and provide this to the CSMS on request.	
Prerequisite(s)	- **ChargingProfileEntries.maxLimit** must be > 2 - The configuration variable **PeriodsPerSchedule** must be > 2	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: *SetChargingProfile* with ChargingProfile 1: **chargingProfilePurpose** is *ChargingStationMaxProfile* **chargingProfileKind** should be *Absolute* **stackLevel** should be *0* **evseId** *0* **startSchedule** *<current dateTime - <Configured max time deviation> seconds>* **numberPhases** *<Configured numberPhases>* **ChargingSchedule**: **duration** *86400* **chargingRateUnit** *<Configured chargingRateUnit>* *Note: If <Configured chargingRateUnit> is W, then the **limit** field will be multiplied by 1000.* **startPeriod** *0*, **limit** *10*	
	ChargingProfile 2: **chargingProfilePurpose** is *TxDefaultProfile* **chargingProfileKind** should be *Absolute* **stackLevel** should be *0* **evseId** *<Configured evseId>* **validFrom** *<current dateTime - <Configured max time deviation> seconds>* **validTo** *<current dateTime + <Configured max time deviation> + 401 seconds>* **startSchedule** *<current dateTime - <Configured max time deviation> seconds>* **numberPhases** *<Configured numberPhases>* **ChargingSchedule**: **duration** *300* **chargingRateUnit** *<Configured chargingRateUnit>* *Note: If <Configured chargingRateUnit> is W, then the **limit** field will be multiplied by 1000.* **startPeriod** *0,60,120,180,260*, **limit** *6,10,8,15,8*	ChargingProfile 3: **chargingProfilePurpose** is *TxProfile* **chargingProfileKind** should be *Absolute* **stackLevel** should be *0* **evseId** *<Configured evseId>* **validFrom** *<current dateTime - <Configured max time deviation> seconds>* **validTo** *<current dateTime + <Configured max time deviation> + 401 seconds>* **startSchedule** *<current dateTime - <Configured max time deviation> seconds>* **numberPhases** *<Configured numberPhases>* **ChargingSchedule**: **duration** *260* **chargingRateUnit** *<Configured chargingRateUnit>* *Note: If <Configured chargingRateUnit> is W, then the **limit** field will be multiplied by 1000.* **startPeriod** *0,50,140,200,240*, **limit** *8,11,16,6,12*
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **GetCompositeScheduleResponse**	**1.** The OCTT sends a **GetCompositeScheduleRequest** with **evseId** *<Configured evseId>* **duration** is *400* **chargingRateUnit** *<Configured chargingRateUnit>*

Test case name	**Get Composite Schedule - Combining chargingProfilePurposes**
Tool validations	* Step 2: (Message: **GetCompositeScheduleResponse**) **status** *Accepted* **evseId** *<Configured evseId>* **ChargingSchedule**: **duration** *400* **chargingRateUnit** *<Configured chargingRateUnit>* *Note: If <Configured chargingRateUnit> is W, then the **limit** field will be multiplied by 1000.* *Note: The period of time between sending the second SetChargingProfileRequest and the **scheduleStart** from the GetCompositeScheduleResponse is called **x**:* **startPeriod** *0*, **limit** *8* **startPeriod** *(50 - x)*, **limit** *10* **startPeriod** *(200 - x)*, **limit** *6* **startPeriod** *(240 - x)*, **limit** *10*
	Post scenario validations: N/a

Table 289. Test Case Id: TC_K_42_CS

Test case name	**Get Composite Schedule - chargingRateUnit not supported**	
Test case Id	TC_K_42_CS	
Use case Id(s)	K08	
Requirement(s)	K08.FR.07	
System under test	Charging Station	
Description	The CSMS requests a composite schedule which is a combination of local limits and the prevailing Charging Profiles of the different chargingProfilePurposes and stack levels.	
Purpose	To verify if the Charging Station is able to reject a GetCompositeScheduleRequest when it asks for a not supported chargingRateUnit.	
Prerequisite(s)	- The Charging Station does NOT support one of the chargingRateUnits; A or W. - The OCTT chargingRateUnit configuration field contains the NOT supported chargingRateUnit.	
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 **GetCompositeScheduleResponse**	**1.** The OCTT sends a **GetCompositeScheduleRequest** with **evseId** *0* **duration** is *300* **chargingRateUnit** *<Configured unsupported chargingRateUnit>*
Tool validations	* Step 2: (Message: **GetCompositeScheduleResponse**) **status** *Rejected* **schedule** is omitted	
	Post scenario validations: N/a	

Table 290. Test Case Id: TC_K_47_CS

Test case name	**Get Composite Schedule - Unknown EVSEId**	
Test case Id	TC_K_47_CS	
Use case Id(s)	K08	
Requirement(s)	K08.FR.05	
System under test	Charging Station	
Description	The CSMS requests a composite schedule which is a combination of local limits and the prevailing Charging Profiles of the different chargingProfilePurposes and stack levels.	
Purpose	To verify if the Charging Station is able to reject a GetCompositeScheduleRequest when it asks for composite schedule for a unknown evseId.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **GetCompositeScheduleResponse**	**1.** The OCTT sends a **GetCompositeScheduleRequest** with **evseId** *<Configured number of evse> + 1* **duration** is *300* **chargingRateUnit** *<Configured chargingRateUnit>*
Tool validations	* Step 2: (Message: **GetCompositeScheduleResponse**) **status** *Rejected* **schedule** is omitted	
	Post scenario validations: N/a	

Table 291. Test Case Id: TC_K_52_CS

Test case name	**Set External Charging Limit (not on a transaction) - ChargingStationExternalConstraints in report**	
Test case Id	TC_K_52_CS	
Use case Id(s)	K12	
Requirement(s)	K12.FR.05	
System under test	Charging Station	
Description	A charging schedule or charging limit has been set by an external system on the Charging Station. Such a charging limit is represented by a charging profile with purpose *ChargingStatioExternalConstraints*.	
Purpose	To verify if the charging station is able to correctly report an external charging limit as *ChargingStationExternalConstraints*.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: An external charging limit has been submitted to Charging Station.	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **GetChargingProfilesResponse**	**1.** The OCTT sends a **GetChargingProfilesRequest** with **chargingProfile.chargingProfilePurpose** *ChargingStationExternalConstraints*
	3. The Charging Station sends a **ReportChargingProfilesRequest**	**4.** The OCTT responds with a **ReportChargingProfilesResponse**
	Note(s): *- If **tbc** is True at Step 3 then step 3 and 4 will be repeated*	
Tool validations	* Step 2: Message **GetChargingProfilesResponse** - **status** *Accepted* * Step 3: Message **ReportChargingProfilesRequest** - **requestId** *Same id as in the request in step 1* - **chargingProfile.chargingProfilePurpose** *ChargingStationExternalConstraints*	
	Post scenario validations: - All report messages have been received and at least one *ChargingStationExternalConstraints* is returned.	

Table 292. Test Case Id: TC_K_53_CS

Test case name	**Charging with load leveling based on High Level Communication - Success**	
Test case Id	TC_K_53_CS	
Use case Id(s)	K15	
Requirement(s)	K15.FR.01,K15.FR.06,K15.FR.09,K15.FR.10	
System under test	Charging Station	
Description	ISO15118-1 E1 AC Charging with load leveling based on High Level Communication, and E4 DC charging with load leveling based on High Level Communication.	
Purpose	To verify if the Charging Station is able to perform load leveling when it receives charging needs from the EV.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): **State is** *Authorized* (local) **State is** *EVConnectedPreSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *RenegotiateChargingLimits*	
Tool validations	**Post scenario validations:** N/a	

Table 293. Test Case Id: TC_K_54_CS

Test case name	**Charging with load leveling based on High Level Communication - No SASchedule (rejected)**	
Test case Id	TC_K_54_CS	
Use case Id(s)	K15, K17	
Requirement(s)	K15.FR.01,K17.FR.04	
System under test	Charging Station	
Description	ISO15118-1 E1 AC Charging with load leveling based on High Level Communication, and E4 DC charging with load leveling based on High Level Communication.	
Purpose	To verify if the Charging Station is able to handle a Rejected status from the CSMS in response to providing the EV charging needs.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): **State is** *Authorized* (local) **State is** *EVConnectedPreSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The Charging Station sends a **NotifyEVChargingNeedsRequest**.	**2.** The OCTT responds with a **NotifyEVChargingNeedsResponse**. With **status** *Rejected*
	3. The Charging Station sends a **NotifyEVChargingScheduleRequest**. Note(s): *- This step is optional. The Charging Station will only send it when the EV returns a charging profile.*	**4.** The OCTT responds with a **NotifyEVChargingScheduleResponse**. With **status** *Accepted*
	5. The Charging Station sends a **TransactionEventRequest**.	**6.** The OCTT responds with a **TransactionEventResponse**.
Tool validations	* Step 1: (Message: **NotifyEVChargingNeedsRequest**) **evseId** *<Configured evseId>* * Step 3: (Message: **NotifyEVChargingScheduleRequest**) **evseId** *<Configured evseId>* * Step 5: Message: **TransactionEventRequest** - **triggerReason** must be *ChargingStateChanged* - **transactionInfo.chargingState** must be *Charging*	
	Post scenario validations: N/a	

Table 294. Test Case Id: TC_K_56_CS

Test case name	**Charging with load leveling based on High Level Communication - Offline**	
Test case Id	TC_K_56_CS	
Use case Id(s)	K15,K17	
Requirement(s)	K15.FR.15,K17.FR.15	
System under test	Charging Station	
Description	ISO15118-1 E1 AC Charging with load leveling based on High Level Communication, and E4 DC charging with load leveling based on High Level Communication.	
Purpose	To verify if the Charging Station is able to perform load leveling when it receives charging needs from the EV and it is offline.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** **RetryBackOffWaitMinimum** is *<Configured RetryBackOffWaitMinimum>*	
	Memory State: *SetChargingProfile* with ChargingProfile: **chargingProfilePurpose** is *TxDefaultProfile* **chargingProfileKind** should be *Absolute* **stackLevel** should be *0* **evseId** *<Configured evseId>* **validFrom** *<current dateTime - <Configured max time deviation> seconds>* **validTo** *<current dateTime + <Configured max time deviation> + 401 seconds>* **startSchedule** *<current dateTime - <Configured max time deviation> seconds>* **numberPhases** *<Configured numberPhases>* **ChargingSchedule**: **duration** *400* **chargingRateUnit** *<Configured chargingRateUnit>* *Note: If <Configured chargingRateUnit> is W, then the **limit** field will be multiplied by 1000.* **startPeriod** *0*, **limit** *6*	
	Reusable State(s): **State is** *Authorized* (local) **State is** *EVConnectedPreSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
		1. *The OCTT closes the WebSocket connection AND does not accept a reconnect.*
		2. *The OCTT accepts the reconnection attempt from the Charging Station, after 90 seconds.*
	3. The Charging Station sends a **NotifyEVChargingScheduleRequest**. Note(s): *- This step is optional.* *- It is allowed to execute this step either before or after the TransactionEventRequest from step 5.*	**4.** The OCTT responds with a **NotifyEVChargingScheduleResponse**. With **status** *Accepted*
	5. The Charging Station sends a **TransactionEventRequest**.	**6.** The OCTT responds with a **TransactionEventResponse**.

Test case name	**Charging with load leveling based on High Level Communication - Offline**
Tool validations	* Step 3: (Message: **NotifyEVChargingScheduleRequest**) **evseId** *<Configured evseId>* * Step 5: Message: **TransactionEventRequest** - **triggerReason** must be *ChargingStateChanged* - **transactionInfo.chargingState** must be *Charging* - **offline** *true*
	Post scenario validations: N/a

Table 295. Test Case Id: TC_K_57_CS

Test case name	**Renegotiating a Charging Schedule - Initiated by EV**	
Test case Id	TC_K_57_CS	
Use case Id(s)	K17	
Requirement(s)	K17.FR.01,K17.FR.06,K17.FR.09,K17.FR.10	
System under test	Charging Station	
Description	The EV signals the Charging Station that it wants to renegotiate and it provides new charging needs, which the Charging Station sends to the CSMS. Based on this and other parameters, the CSMS calculates a new charging schedule and sends it via SetChargingProfileRequest to Charging Station, which communicates it to the EV.	
Purpose	To verify if the Charging Station is able to perform load leveling when it receives a renegotiate request from the EV.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): **State** is *Authorized* (local) **State** is *EVConnectedPreSession* **State** is *RenegotiateChargingLimits*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The Charging Station sends a **NotifyEVChargingNeedsRequest**.	**2.** The OCTT responds with a **NotifyEVChargingNeedsResponse**. With **status** *Accepted*
	4. The Charging Station responds with a **SetChargingProfileResponse**	**3.** The OCTT sends a **SetChargingProfileRequest** with **chargingProfile.chargingProfilePurpose** *TxProfile* **chargingProfile.transactionId** *<Provided transactionId from before>* **chargingProfile.chargingSchedule[0].chargingRateUnit** *<Configured chargingRateUnit>* *Note: If <Configured chargingRateUnit> is W, then the **limit** field will be multiplied by 1000.* **chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].startPeriod** *0*, **chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].limit** *15*
	5. The Charging Station sends a **NotifyEVChargingScheduleRequest**. Note(s): *- This step is optional. The Charging Station will only send it when the EV returns a charging profile.*	**6.** The OCTT responds with a **NotifyEVChargingScheduleResponse**. With **status** *Accepted*
	7. The Charging Station sends a **TransactionEventRequest**. Note(s): *- This step is optional. But the Charging Station will probably send it, otherwise it would not have renegotiated.*	**8.** The OCTT responds with a **TransactionEventResponse**.

Test case name	**Renegotiating a Charging Schedule - Initiated by EV**
Tool validations	* Step 1: (Message: **NotifyEVChargingNeedsRequest**) **evseId** *<Configured evseId>* * Step 4: (Message: **SetChargingProfileResponse**) **status** *Accepted* * Step 5: (Message: **NotifyEVChargingScheduleRequest**) **evseId** *<Configured evseId>* * Step 7: Message: **TransactionEventRequest** - **triggerReason** must be *ChargingRateChanged*
	Post scenario validations: N/a

Table 296. Test Case Id: TC_K_58_CS

Test case name	**Renegotiating a Charging Schedule - Initiated by CSMS**	
Test case Id	TC_K_58_CS	
Use case Id(s)	K17	
Requirement(s)	K17.FR.01,K17.FR.06,K17.FR.09,K17.FR.10	
System under test	Charging Station	
Description	The EV signals the Charging Station that it wants to renegotiate and it provides new charging needs, which the Charging Station sends to the CSMS. Based on this and other parameters, the CSMS calculates a new charging schedule and sends it via SetChargingProfileRequest to Charging Station, which communicates it to the EV.	
Purpose	To verify if the Charging Station is able to perform load leveling when it receives a renegotiate request from the CSMS.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): **State** is *Authorized* (local) **State** is *EVConnectedPreSession* **State** is *RenegotiateChargingLimits*	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SetChargingProfileResponse**	**1.** The OCTT sends a **SetChargingProfileRequest** with **chargingProfile.chargingProfilePurpose** *TxProfile* **chargingProfile.transactionId** *<Provided transactionId from before>* **chargingProfile.chargingSchedule[0].chargingRateUnit** *<Configured chargingRateUnit>* **chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].startPeriod** *0* If <Configured chargingRateUnit> is W: **chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].limit** *8000* Else: **chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].limit** *8*
	3. The Charging Station sends a **NotifyEVChargingScheduleRequest**. Note(s): *- This step is optional. The Charging Station will only send it when the EV returns a charging profile.*	**4.** The OCTT responds with a **NotifyEVChargingScheduleResponse**. With **status** *Accepted*
	5. The Charging Station sends a **TransactionEventRequest**. Note(s): *- This step is optional. But the Charging Station will send it, when it was charging above a limit of 8/8000.*	**6.** The OCTT responds with a **TransactionEventResponse**.

Test case name	**Renegotiating a Charging Schedule - Initiated by CSMS**
Tool validations	* Step 1: (Message: **NotifyEVChargingNeedsRequest**) **evseId** *<Configured evseId>* * Step 4: (Message: **SetChargingProfileResponse**) **status** *Accepted* * Step 5: (Message: **NotifyEVChargingScheduleRequest**) **evseId** *<Configured evseId>* * Step 7: Message: **TransactionEventRequest** - **triggerReason** must be *ChargingRateChanged*
	Post scenario validations: N/a

2.13. L Firmware Management

Table 297. 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.
	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 - Installation successful	
	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 298. 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 299. 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 300. Test Case Id: TC_L_05_CS

Test case name	**Secure Firmware Update - InvalidCertificate**	
Test case Id	TC_L_05_CS	
Use case Id(s)	L01	
Requirement(s)	L01.FR.01,L01.FR.02,L01.FR.10,L01.FR.20,L01.FR.21,L01.FR.22	
System under test	Charging Station	
Description	The CSMS is able to request the Charging Station to securely download and install a new firmware by sending an UpdateFirmwareRequest with a signingCertificate.	
Purpose	To verify if the Charging Station is able to identify it receiving an invalid signing certificate and report this to the CSMS.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** <Generated Invalid Firmware SigningCertificate> should be a trusted certificate and not be the same as the <Configured Valid Firmware SigningCertificate>	
	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** *<Generated invalid firmware signingCertificate>* **firmware.signature** *<Configured signature>*
	3. The Charging Station sends a **SecurityEventNotificationRequest**. With **type** *InvalidFirmwareSigningCertificate*	**4.** The OCTT responds with a **SecurityEventNotificationResponse**.
Tool validations	* Step 2: Message **UpdateFirmwareResponse** - **status** *InvalidCertificate* OR *RevokedCertificate* * Step 3: Message **SecurityEventNotificationRequest** - **type** *InvalidFirmwareSigningCertificate*	
	Post scenario validations: N/a	

Table 301. 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 302. 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 303. 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**
	Note: *Step 14 through 17 can be send in a different order.*	
	14. 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 the reboot sequence at step 11) yet.* *- And if the Charging Station did not become inoperative after the firmware update failure. It is recommended for a Charging Station to fallback to the previous firmware after a firmware update failure.*	**15.** The OCTT responds accordingly.
	16. The Charging Station sends a **FirmwareStatusNotificationRequest** With **status** *InstallVerificationFailed* or *InstallationFailed*	**17.** The OCTT responds with a **FirmwareStatusNotificationResponse**

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 14: 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 16: Message **FirmwareStatusNotificationRequest** - **status** *InstallVerificationFailed* or *InstallationFailed*
	Post scenario validations: N/a

Table 304. 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 305. 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 306. 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 307. 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 308. Test Case Id: TC_L_14_CS

Test case name	**Secure Firmware Update - Unable to install and activate 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/activate a new firmware by sending an UpdateFirmwareRequest with a signingCertificate. When the *Installing* phase is not possible while a transaction is ongoing, Charging Station will report *InstallScheduled* and wait for transaction(s) to finish first, else it will immediately report *Installing*. In both cases before activation of new firmware by (optional) reboot and a reconnect, Charging Station will always wait for transaction(s) to finish.
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 and/or activate 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 and activate 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* or **status** *Installing* Note(s): *- InstallScheduled only applies when Charging Station is not able to install while a transaction is active.*	**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.*	
	11a. If Charging Station reported *Installing* in step 9 then wait a while (30-60 s) before continuing with next steps to stop transactions to allow time to install firmware.	
	12. Execute **Reusable State** *ParkingBayUnoccupied* for *<Configured connectorId>* Note(s): *- The Charging Station will proceed to this end state. This will cause the first 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 second transaction to stop.* *- The Charging Station will start the firmware update process (if it had not started installing in step 9) the moment this second transaction ends or when all interactions with the EV Driver are done (so after the cable has been unplugged, assuming 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 want to set its connectors to Unavailable, before proceeding installing the new firmware.*	**15.** The OCTT responds accordingly.

Test case name	**Secure Firmware Update - Unable to install and activate firmware with ongoing transaction - AllowNewSessionsPendingFirmwareUpdate is true**	
	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 report Installing at step 9 and did not reboot before firmware installation, at step 16 (because that step already reports Installing).*	**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 sent 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 and activate 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* or *Installing* * Step 14: (optional) 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: (optional depending on step 9) 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 309. Test Case Id: TC_L_15_CS

Test case name	**Secure Firmware Update - Unable to install and activate 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. When the *Installing* phase is not possible while a transaction is ongoing, Charging Station will report *InstallScheduled* and wait for transaction(s) to finish first, else it will immediately report *Installing*. In both cases before activation of new firmware by (optional) reboot and a reconnect, Charging Station will always wait for transaction(s) to finish.
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 and/or activate firmware while there is an ongoing transaction.
Before (Preparations)	**Configuration State:** **AllowNewSessionsPendingFirmwareUpdate** is *false*
	Memory State: N/a
	Reusable State(s): **State is** *EnergyTransferStarted*

Test case name	**Secure Firmware Update - Unable to install and activate firmware with ongoing transaction - AllowNewSessionsPendingFirmwareUpdate is false**	
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* or **status** *Installing* Note: *InstallScheduled only applies when Charging Station is not able to install while a transaction is active. 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 that its Available connector(s) have been set to Unavailable. Note(s): *- This step needs to be executed for all connectors with AvailabilityState Available.*	**12.** The OCTT responds accordingly.
	12a. If Charging Station reported *Installing* in step 9 then wait a while (30-60 s) before continuing with next steps to stop transaction to allow time to install firmware.	
	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 (if it had not started installing in step 9) the moment the transaction ends or when all interactions with the EV Driver are done (so after the cable has been unplugged, assuming 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 want to set its last connector to Unavailable, before proceeding installing the new firmware.*	**15.** The OCTT responds accordingly.

Test case name	Secure Firmware Update - Unable to install and activate firmware with ongoing transaction - AllowNewSessionsPendingFirmwareUpdate is false	
	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 report Installing at step 9 and did not reboot before firmware installation, at step 16 (because that step already reports Installing).*	**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 sent 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 and activate 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* or *Installing* * 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: (optional) 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: (optional depending on step 9) 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 310. 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 311. Test Case Id: TC_L_18_CS

Test case name	**Secure Firmware Update - Missing firmware signing certificate and signature**	
Test case Id	TC_L_18_CS	
Use case Id(s)	L01	
Requirement(s)	N/a	
System under test	Charging Station	
Description	The CSMS is able to request the Charging Station to securely download and install a new firmware by sending an UpdateFirmwareRequest with a signingCertificate.	
Purpose	To verify if the Charging Station is not accepting a non-secure firmware update request, when supporting secure firmware update.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **UpdateFirmwareResponse**	**1.** The OCTT sends a **UpdateFirmwareRequest** with **firmware.installDateTime** *<Current DateTime - 2 hours>* **firmware.location** *<Configured firmware_location>* **firmware.retrieveDateTime** *<Current DateTime - 2 hours>* **firmware.signingCertificate** is omitted **firmware.signature** is omitted
Tool validations	* Step 2: Message **UpdateFirmwareResponse** - **status** *Rejected* OR *InvalidCertificate*	
	Post scenario validations: N/a	

2.14. M ISO 15118 CertificateManagement

Table 312. Test Case Id: TC_M_01_CS

Test case name	**Install CA certificate - CSMSRootCertificate**	
Test case Id	TC_M_01_CS	
Use case Id(s)	M05	
Requirement(s)	M05.FR.01,M05.FR.02	
System under test	Charging Station	
Description	The CSMS is able to request the Charging Station to install new Root CA certificates using the InstallCertificateRequest message.	
Purpose	To verify if the Charging Station is able to install a new CSMSRootCertificate.	
Prerequisite(s)	- The Charging Station supports Security Profile 2 or 3.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *CertificateInstalled* for certificateType *CSMSRootCertificate* (Root 2) Note(s): *- When the Charging Station has the following configuration; **AdditionalRootCertificateCheck** implemented with value **true**, then a custom CSMSRootCertificate should be used.*	
	2. Execute **Reusable State** *GetInstalledCertificates* for certificateType *CSMSRootCertificate*	
Tool validations	N/a	
	Post scenario validations: N/a	

Table 313. Test Case Id: TC_M_02_CS

Test case name	**Install CA certificate - ManufacturerRootCertificate**	
Test case Id	TC_M_02_CS	
Use case Id(s)	M05	
Requirement(s)	M05.FR.01,M05.FR.02	
System under test	Charging Station	
Description	The CSMS is able to request the Charging Station to install new Root CA certificates using the InstallCertificateRequest message.	
Purpose	To verify if the Charging Station is able to install a new ManufacturerRootCertificate.	
Prerequisite(s)	The Charging Station supports **signed** firmware updates.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *CertificateInstalled* for certificateType *ManufacturerRootCertificate*	
	2. Execute **Reusable State** *GetInstalledCertificates* for certificateType *ManufacturerRootCertificate*	
Tool validations	N/a	
	Post scenario validations: N/a	

Table 314. Test Case Id: TC_M_03_CS

Test case name	**Install CA certificate - V2GRootCertificate**	
Test case Id	TC_M_03_CS	
Use case Id(s)	M05	
Requirement(s)	M05.FR.01,M05.FR.02	
System under test	Charging Station	
Description	The CSMS is able to request the Charging Station to install new Root CA certificates using the InstallCertificateRequest message.	
Purpose	To verify if the Charging Station is able to install a new V2GRootCertificate.	
Prerequisite(s)	- The Charging Station supports ISO 15118. - The Charging Station does NOT have the following configuration; **AdditionalRootCertificateCheck** is implemented with value *true*	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *CertificateInstalled* for certificateType *V2GRootCertificate*	
	2. Execute **Reusable State** *GetInstalledCertificates* for certificateType *V2GRootCertificate*	
Tool validations	N/a	
	Post scenario validations: N/a	

Table 315. Test Case Id: TC_M_04_CS

Test case name	**Install CA certificate - MORootCertificate**	
Test case Id	TC_M_04_CS	
Use case Id(s)	M05	
Requirement(s)	M05.FR.01,M05.FR.02	
System under test	Charging Station	
Description	The CSMS is able to request the Charging Station to install new Root CA certificates using the InstallCertificateRequest message.	
Purpose	To verify if the Charging Station is able to install a new MORootCertificate.	
Prerequisite(s)	- The Charging Station supports ISO 15118. - The Charging Station does NOT have the following configuration; **AdditionalRootCertificateCheck** is implemented with value *true*	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *CertificateInstalled* for certificateType *MORootCertificate*	
	2. Execute **Reusable State** *GetInstalledCertificates* for certificateType *MORootCertificate*	
Tool validations	N/a	
	Post scenario validations: N/a	

Table 316. Test Case Id: TC_M_07_CS

Test case name	**Install CA certificate - Rejected - Certificate invalid**	
Test case Id	TC_M_07_CS	
Use case Id(s)	M05	
Requirement(s)	M05.FR.01,M05.FR.07	
System under test	Charging Station	
Description	The CSMS is able to request the Charging Station to install new Root CA certificates using the InstallCertificateRequest message.	
Purpose	To verify if the Charging Station is able to reject an invalid certificate.	
Prerequisite(s)	- The Charging Station supports Security Profile 2 or 3. - The Charging Station does NOT have the following configuration; **AdditionalRootCertificateCheck** is implemented with value *true*	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **InstallCertificateResponse**	**1.** The OCTT sends a **InstallCertificateRequest** with **certificateType** is *CSMSRootCertificate* **certificate** is *<Generated Expired Certificate>*
	4. The Charging Station responds with a **GetInstalledCertificateIdsResponse**	**3.** The OCTT sends a **GetInstalledCertificateIdsRequest** with **certificateType** is *CSMSRootCertificate*
Tool validations	* Step 2: Message: **InstallCertificateResponse** - **status** must be *Rejected* * Step 4: Message: **GetInstalledCertificateIdsResponse** - **status** must be *Accepted* - **certificateHashDataChain** must NOT contain an entry with following values: *Note: Order does not matter.* - **certificateType** is *CSMSRootCertificate* - **certificateHashData** contains *<HashData from configured new CSMS Root certificate>*	
	Post scenario validations: N/a	

Table 317. Test Case Id: TC_M_09_CS

Test case name	**Install CA certificate - AdditionalRootCertificateCheck - Rejected**	
Test case Id	TC_M_09_CS	
Use case Id(s)	M05	
Requirement(s)	M05.FR.10,M05.FR.11	
System under test	Charging Station	
Description	The CSMS is able to request the Charging Station to install new Root CA certificates using the InstallCertificateRequest message.	
Purpose	To verify if the Charging Station is able to reject installing a new CSMSRootCertificate that is not signed by the old CSMSRootCertificate, while additional security measures for installing a root certificate is active.	
Prerequisite(s)	- The Charging Station supports Security Profile 2 or 3. - The Charging Station has the configuration variable **AdditionalRootCertificateCheck** implemented with value *true*	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **InstallCertificateResponse**	**1.** The OCTT sends a **InstallCertificateRequest** with **certificateType** is *CSMSRootCertificate* **certificate** is *<Configured CSMSRootCertificate>* Note(s): *- CSMSRootCertificate must have not been signed by old certificate.*
	4. The Charging Station responds with a **GetInstalledCertificateIdsResponse**	**3.** The OCTT sends a **GetInstalledCertificateIdsRequest** with **certificateType** is *CSMSRootCertificate*
Tool validations	* Step 2: Message: **InstallCertificateResponse** - **status** must be *Rejected* * Step 4: Message: **GetInstalledCertificateIdsResponse** - **status** must be *Accepted* - **certificateHashDataChain** must contain one entry with following values: - **certificateType** is *CSMSRootCertificate* - **certificateHashData** contains *<HashData from configured old CSMS Root certificate>*	
	Post scenario validations: N/a	

Table 318. Test Case Id: TC_M_30_CS

Test case name	**Install CA certificate - AdditionalRootCertificateCheck - Reconnect using new CSMS Root - Success**	
Test case Id	TC_M_30_CS	
Use case Id(s)	M05	
Requirement(s)	M05.FR.13	
System under test	Charging Station	
Description	The CSMS is able to request the Charging Station to install new Root CA certificates using the InstallCertificateRequest message.	
Purpose	To verify if the Charging Station is able to reconnect to the CSMS, while using a new CSMS Root certificate.	
Prerequisite(s)	- The Charging Station supports Security Profile 2 or 3. - The Charging Station has the configuration variable **AdditionalRootCertificateCheck** implemented with value *true* - The at the OCTT configured new CSMSRootCertificate must be signed by the old CSMS Root certificate.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: *CertificateInstalled* for certificateType *CSMSRootCertificate* and certificate *<Configured new CSMS Root certificate 2>* If security profile 3 is enabled, then: *RenewChargingStationCertificate* for certificateType *ChargingStationCertificate*	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **ResetResponse**	**1.** The OCTT sends a **ResetRequest** with **type** *OnIdle*
	4. During the TLS handshake the Charging Station validates the CSMS certificate. Note(s): - *This connection attempt must succeed.*	**3.** During the TLS handshake the OCTT provides a CSMS certificate which is signed by the *<Configured new CSMS Root certificate>*
	5. Execute **Reusable State** *Booted*	
	7. The Charging Station responds with a **GetInstalledCertificateIdsResponse**	**6.** The OCTT sends a **GetInstalledCertificateIdsRequest** with **certificateType** is *CSMSRootCertificate*
Tool validations	* Step 2: Message **ResetResponse** - **status** *Accepted* * Step 7: Message: **GetInstalledCertificateIdsResponse** - **status** must be *Accepted* - **certificateHashDataChain** must NOT contain an entry with following values: - **certificateType** is *CSMSRootCertificate* - **certificateHashData** contains *<HashData from configured old CSMS Root certificate>*	
	Post scenario validations: - N/a	

Table 319. Test Case Id: TC_M_31_CS

Test case name	**Install CA certificate - AdditionalRootCertificateCheck - Reconnect using new CSMS Root - Fallback mechanism**	
Test case Id	TC_M_31_CS	
Use case Id(s)	M05	
Requirement(s)	M05.FR.14	
System under test	Charging Station	
Description	The CSMS is able to request the Charging Station to install new Root CA certificates using the InstallCertificateRequest message.	
Purpose	To verify if the Charging Station is able to reconnect to the CSMS using the old CSMS Root certificate, when validating the CSMS certificate using the new CSMS Root certificate fails.	
Prerequisite(s)	- The Charging Station supports Security Profile 2 or 3. - The Charging Station has the configuration variable **AdditionalRootCertificateCheck** implemented with value *true* - The at the OCTT configured new CSMSRootCertificate must be signed by the old CSMS Root certificate.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: *CertificateInstalled* for certificateType *CSMSRootCertificate* and certificate *<Configured (new) CSMS Root certificate 2>*	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **ResetResponse**	**1.** The OCTT sends a **ResetRequest** with **type** *OnIdle*
	4. During the TLS handshake the Charging Station validates the CSMS certificate. Note(s): *- This connection attempt fails, because the Charging Station will use the new CSMS Root certificate to validate the CSMS certificate.*	**3.** During the TLS handshake the OCTT provides a CSMS certificate which is signed by the *<Configured old CSMS Root certificate>*
	5. The Charging Station re-validates the CSMS certificate. Note(s): *- This connection attempt succeeds, because the Charging Station will now use the old CSMS Root certificate to validate the CSMS certificate.*	
	6. Execute **Reusable State** *Booted*	
	8. The Charging Station responds with a **GetInstalledCertificateIdsResponse**	**7.** The OCTT sends a **GetInstalledCertificateIdsRequest** with **certificateType** is *CSMSRootCertificate*

Test case name	**Install CA certificate - AdditionalRootCertificateCheck - Reconnect using new CSMS Root - Fallback mechanism**
Tool validations	* Step 2: Message **ResetResponse** - **status** *Accepted* * Step 8: Message: **GetInstalledCertificateIdsResponse** - **status** must be *Accepted* - **certificateHashDataChain** must contain an entry with following values: - **certificateType** is *CSMSRootCertificate* - **certificateHashData** contains *<HashData from configured old CSMS Root certificate>*
	Post scenario validations: - N/a

Table 320. Test Case Id: TC_M_12_CS

Test case name	**Retrieve certificates from Charging Station - CSMSRootCertificate**	
Test case Id	TC_M_12_CS	
Use case Id(s)	M03	
Requirement(s)	M03.FR.01,M03.FR.03,M03.FR.04	
System under test	Charging Station	
Description	The CSMS is able to retrieve the certificates installed at the Charging Station using the GetInstalledCertificateIdsRequest message.	
Purpose	To verify if the Charging Station is able to provide the hashData from all stored CSMSRootCertificates.	
Prerequisite(s)	- The Charging Station supports Security Profile 2 or 3.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: *CertificateInstalled* from certificateType *CSMSRootCertificate*	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *GetInstalledCertificates* for certificateType *CSMSRootCertificate*	
Tool validations	N/a	
	Post scenario validations: N/a	

Table 321. Test Case Id: TC_M_13_CS

Test case name	**Retrieve certificates from Charging Station - ManufacturerRootCertificate**	
Test case Id	TC_M_13_CS	
Use case Id(s)	M03	
Requirement(s)	M03.FR.01,M03.FR.03,M03.FR.04	
System under test	Charging Station	
Description	The CSMS is able to retrieve the certificates installed at the Charging Station using the GetInstalledCertificateIdsRequest message.	
Purpose	To verify if the Charging Station is able to provide the hashData from all stored ManufacturerRootCertificate.	
Prerequisite(s)	- The Charging Station supports **signed** firmware updates.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: *CertificateInstalled* from certificateType *ManufacturerRootCertificate*	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *GetInstalledCertificates* for certificateType *ManufacturerRootCertificate*	
Tool validations	N/a	
	Post scenario validations: N/a	

Table 322. Test Case Id: TC_M_14_CS

Test case name	**Retrieve certificates from Charging Station - V2GRootCertificate**	
Test case Id	TC_M_14_CS	
Use case Id(s)	M03	
Requirement(s)	M03.FR.01,M03.FR.03,M03.FR.04	
System under test	Charging Station	
Description	The CSMS is able to retrieve the certificates installed at the Charging Station using the GetInstalledCertificateIdsRequest message.	
Purpose	To verify if the Charging Station is able to provide the hashData from all stored V2GRootCertificate.	
Prerequisite(s)	The Charging Station supports ISO 15118.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: *CertificateInstalled* from certificateType *V2GRootCertificate*	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *GetInstalledCertificates* for certificateType *V2GRootCertificate*	
Tool validations	N/a	
	Post scenario validations: N/a	

Table 323. Test Case Id: TC_M_15_CS

Test case name	**Retrieve certificates from Charging Station - V2GCertificateChain**	
Test case Id	TC_M_15_CS	
Use case Id(s)	M03	
Requirement(s)	M03.FR.01,M03.FR.03,M03.FR.04,M03.FR.05	
System under test	Charging Station	
Description	The CSMS is able to retrieve the certificates installed at the Charging Station using the GetInstalledCertificateIdsRequest message.	
Purpose	To verify if the Charging Station is able to provide the hashData from all stored certificates that are part of a V2GCertificateChain.	
Prerequisite(s)	- The Charging Station supports ISO 15118. - The Charging Station has atleast one V2GCertificateChain installed.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *GetInstalledCertificates* for certificateType *V2GCertificateChain*	
Tool validations	* Step 1: Message: **GetInstalledCertificateIdsResponse** - **status** must be *Accepted* - **certificateHashDataChain** must contain an entry with following values: *Note: Order does not matter.* - **certificateType** is *V2GCertificateChain* - **certificateHashData** uses the childCertificateHashData field	
	Post scenario validations: N/a	

Table 324. Test Case Id: TC_M_16_CS

Test case name	**Retrieve certificates from Charging Station - MORootCertificate**	
Test case Id	TC_M_16_CS	
Use case Id(s)	M03	
Requirement(s)	M03.FR.01,M03.FR.03,M03.FR.04	
System under test	Charging Station	
Description	The CSMS is able to retrieve the certificates installed at the Charging Station using the GetInstalledCertificateIdsRequest message.	
Purpose	To verify if the Charging Station is able to provide the hashData from all stored MORootCertificate.	
Prerequisite(s)	The Charging Station supports ISO 15118.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: *CertificateInstalled* from certificateType *MORootCertificate*	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *GetInstalledCertificates* for certificateType *MORootCertificate*	
Tool validations	N/a	
	Post scenario validations: N/a	

Table 325. Test Case Id: TC_M_17_CS

Test case name	**Retrieve certificates from Charging Station - CSMSRootCertificate & ManufacturerRootCertificate**	
Test case Id	TC_M_17_CS	
Use case Id(s)	M03	
Requirement(s)	M03.FR.01,M03.FR.03,M03.FR.04	
System under test	Charging Station	
Description	The CSMS is able to retrieve the certificates installed at the Charging Station using the GetInstalledCertificateIdsRequest message.	
Purpose	To verify if the Charging Station is able to provide the hashData from all stored CSMSRootCertificates and ManufacturerRootCertificates	
Prerequisite(s)	- The Charging Station supports Security Profile 2 or 3. - The Charging Station supports **signed** firmware updates.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: *CertificateInstalled* from certificateType *CSMSRootCertificate* *CertificateInstalled* from certificateType *ManufacturerRootCertificate*	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *GetInstalledCertificates* for certificateType *CSMSRootCertificate* AND *ManufacturerRootCertificate*	
Tool validations	N/a	
	Post scenario validations: N/a	

Table 326. Test Case Id: TC_M_18_CS

Test case name	**Retrieve certificates from Charging Station - All certificateTypes**	
Test case Id	TC_M_18_CS	
Use case Id(s)	M03	
Requirement(s)	M03.FR.01,M03.FR.03,M03.FR.04	
System under test	Charging Station	
Description	The CSMS is able to retrieve the certificates installed at the Charging Station using the GetInstalledCertificateIdsRequest message.	
Purpose	To verify if the Charging Station is able to provide the hashData from all stored certificates	
Prerequisite(s)	- The Charging Station supports Security Profile 2 or 3. - The Charging Station supports **signed** firmware updates.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: *CertificateInstalled* from certificateType *CSMSRootCertificate* *CertificateInstalled* from certificateType *ManufacturerRootCertificate*	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **GetInstalledCertificateIdsResponse**	**1.** The OCTT sends a **GetInstalledCertificateIdsRequest** With **certificateType** is omitted.
Tool validations	* Step 2: Message: **GetInstalledCertificateIdsResponse** - **status** must be *Accepted* - **certificateHashDataChain** must contain the following two entries with following values: *Note: Order does not matter.* **Entry 1:** - **certificateHashDataChain[0].certificateType** is *CSMSRootCertificate* - **certificateHashDataChain[0].certificateHashData** contains *<HashData from configured new CSMS Root certificate>* **Entry 2:** - **certificateHashDataChain[1].certificateType** is *ManufacturerRootCertificate* - **certificateHashDataChain[1].certificateHashData** contains *<HashData from configured new Manufacturer Root certificate>*	
	Post scenario validations: N/a	

Table 327. Test Case Id: TC_M_19_CS

Test case name	**Retrieve certificates from Charging Station - No matching certificate found**	
Test case Id	TC_M_19_CS	
Use case Id(s)	M03	
Requirement(s)	M03.FR.01,M03.FR.02	
System under test	Charging Station	
Description	The CSMS is able to retrieve the certificates installed at the Charging Station using the GetInstalledCertificateIdsRequest message.	
Purpose	To verify if the Charging Station is able to respond that it did not find any certificate of the requested certificateType.	
Prerequisite(s)	The Charging Station does not have a MORootCertificate installed, or it must be possible to remove it.	
Before (Preparations)	**Configuration State:** OCTT checks to make sure that no MORootCertificate is installed via GetInstalledCertificateIds. If an MORootCertificate exists it removes it via DeleteCertificate.	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **GetInstalledCertificateIdsResponse**	**1.** The OCTT sends a **GetInstalledCertificateIdsRequest** With **certificateType** is *MORootCertificate*
Tool validations	* Step 2: Message: **GetInstalledCertificateIdsResponse** - **status** must be *NotFound* - **certificateHashDataChain** must be omitted.	
	Post scenario validations: N/a	

Table 328. Test Case Id: TC_M_20_CS

Test case name	**Delete a certificate from a Charging Station - Success**	
Test case Id	TC_M_20_CS	
Use case Id(s)	M04	
Requirement(s)	M04.FR.01,M04.FR.02	
System under test	Charging Station	
Description	The CSMS is able to request the Charging Station to delete an installed certificate using the DeleteCertificateRequest message.	
Purpose	To verify if the Charging Station is able to delete an installed certificate.	
Prerequisite(s)	- The Charging Station supports Security Profile 2 or 3.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): *GetInstalledCertificates* with certificateType *CSMSRootCertificate* *CertificateInstalled* with certificateType *CSMSRootCertificate* (When no certificate is returned at *GetInstalledCertificates*)	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *GetInstalledCertificates* with certificateType *CSMSRootCertificate*	
	3. The Charging Station responds with a **DeleteCertificateResponse**	**2.** The OCTT sends a **DeleteCertificateRequest** with **certificateHashData** contains *<Returned certificateHashData at step 1>*
	4. Execute **Reusable State** *GetInstalledCertificates* with certificateType *CSMSRootCertificate*	
Tool validations	* Step 1: - Certificate that is going to be deleted is present. * Step 3: Message: **DeleteCertificateResponse** - **status** must be *Accepted* * Step 4: - Certificate that should be deleted is not present anymore.	
	Post scenario validations: N/a	

Table 329. Test Case Id: TC_M_22_CS

Test case name	**Delete a certificate from a Charging Station - No matching certificate found**	
Test case Id	TC_M_22_CS	
Use case Id(s)	M04	
Requirement(s)	M04.FR.01,M04.FR.04	
System under test	Charging Station	
Description	The CSMS is able to request the Charging Station to delete an installed certificate using the DeleteCertificateRequest message.	
Purpose	To verify if the Charging Station is able to respond that no certificate is installed that matches the provided certificateHashData.	
Prerequisite(s)	- The Charging Station supports Security Profile 2 or 3.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *GetInstalledCertificates* with certificateType *CSMSRootCertificate*.	
	3. The Charging Station responds with a **DeleteCertificateResponse**	**2.** The OCTT sends a **DeleteCertificateRequest** with **certificateHashData** is *<certificateHashData from unknown certificate>*
Tool validations	* Step 3: Message: **DeleteCertificateResponse** - **status** must be *NotFound*	
	Post scenario validations: N/a	

Table 330. Test Case Id: TC_M_23_CS

Test case name	**Delete a certificate from a Charging Station - Unable to delete the Charging Station Certificate**	
Test case Id	TC_M_23_CS	
Use case Id(s)	M04	
Requirement(s)	M04.FR.01,M04.FR.06	
System under test	Charging Station	
Description	The CSMS is able to request the Charging Station to delete an installed certificate using the DeleteCertificateRequest message.	
Purpose	To verify if the Charging Station does NOT allow the deletion of the Charging Station certificate.	
Prerequisite(s)	- The Charging Station supports Security Profile 3. - A valid *CSMSRootCertificate* is installed on the Charging Station.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): *RenewChargingStationCertificate* for certificateType *ChargingStationCertificate*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *GetInstalledCertificates* with certificateType *omitted*.	
	3. The Charging Station responds with a **DeleteCertificateResponse**	**2.** The OCTT sends a **DeleteCertificateRequest** with **certificateHashData** is *<certificateHashData from the generated ChargingStationCertificate at before.>*
Tool validations	* Step 3: Message: **DeleteCertificateResponse** - **status** must be *NotFound* OR *Failed*	
	Post scenario validations: N/a	

Table 331. Test Case Id: TC_M_24_CS

Test case name	**Get Charging Station Certificate status - Success**	
Test case Id	TC_M_24_CS	
Use case Id(s)	M06	
Requirement(s)	M06.FR.06,M06.FR.07	
System under test	Charging Station	
Description	The Charging Station is able to request the CSMS to get the status of a (V2G) Charging Station certificate.	
Purpose	To verify if the Charging Station is able to request the status of a (V2G) Charging Station certificate.	
Prerequisite(s)	- The Charging Station supports ISO 15118.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: *CertificateInstalled* from certificateType *V2GRootCertificate* *CertificateInstalled* from certificateType *MORootCertificate* *RenewV2GChargingStationCertificate*	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The Charging Station sends a **GetCertificateStatusRequest**	**2.** The OCTT responds with a **GetCertificateStatusResponse** with **status** *Accepted* **ocspResult** *<OCSPResponse class as defined in IETF RFC 6960. DER encoded (as defined in IETF RFC 6960), and then base64 encoded.>*
Tool validations	N/a	
	Post scenario validations: N/a	

Table 332. Test Case Id: TC_M_25_CS

Test case name	**Get Charging Station Certificate status - Rejected**	
Test case Id	TC_M_25_CS	
Use case Id(s)	M06	
Requirement(s)	M06.FR.04	
System under test	Charging Station	
Description	The Charging Station is able to request the CSMS to get the status of a (V2G) Charging Station certificate.	
Purpose	To verify if the Charging Station is able to handle receiving a rejected status after requesting the status of a (V2G) Charging Station certificate.	
Prerequisite(s)	- The Charging Station supports ISO 15118.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: *CertificateInstalled* from certificateType *V2GRootCertificate* *CertificateInstalled* from certificateType *MORootCertificate* *RenewV2GChargingStationCertificate*	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The Charging Station sends a **GetCertificateStatusRequest**	**2.** The OCTT responds with a **GetCertificateStatusResponse** with **status** *Failed* **ocspResult** is omitted.
Tool validations	N/a	
	Post scenario validations: N/a	

Table 333. Test Case Id: TC_M_26_CS

Test case name	**Certificate Installation EV - Success**	
Test case Id	TC_M_26_CS	
Use case Id(s)	M01	
Requirement(s)	M01.FR.01	
System under test	Charging Station	
Description	The EV initiates installing a new certificate. The Charging Station forwards the request for a new certificate to the CSMS.	
Purpose	To verify if the Charging Station is able to forward the request to the CSMS.	
Prerequisite(s)	- The Charging Station supports ISO 15118.	
Before (Preparations)	**Configuration State:** -The test case calls *SendISO15118AuthorizationMethod* method with parameter *PnC* in order to inform the EV emulator about the expected authorization method. -The test case calls *SendInstallISO15118CertificateMethod* method in order to trigger the EV emulator to initiate installing a new certificate.	
	Memory State: *CertificateInstalled* from certificateType *V2GRootCertificate* *CertificateInstalled* from certificateType *MORootCertificate*	
	Reusable State(s): **State** is *EVConnectedPreSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The Charging Station sends a **Get15118EVCertificateRequest**	**2.** The OCTT responds with a **Get15118EVCertificateResponse** with **status** *Accepted* **exiResponse** *<Raw CertificateInstallationRes response for the EV, Base64 encoded.>*
	3. The Charging Station sends an **AuthorizeRequest**	**4.** The OCTT responds with an **AuthorizeResponse** with **status** *Accepted*
Tool validations	* Step 1: Message: **Get15118EVCertificateRequest** - **action** must be *Install*	
	Post scenario validations: N/a	

Table 334. Test Case Id: TC_M_27_CS

Test case name	**Certificate Installation EV - Failed**	
Test case Id	TC_M_27_CS	
Use case Id(s)	M01	
Requirement(s)	N/a	
System under test	Charging Station	
Description	The EV initiates installing a new certificate. The Charging Station forwards the request for a new certificate to the CSMS.	
Purpose	To verify if the Charging Station is able to handle receiving a Failed status.	
Prerequisite(s)	- The Charging Station supports ISO 15118.	
Before (Preparations)	**Configuration State:** -The test case calls *SendISO15118AuthorizationMethod* method with parameter *PnC* in order to inform the EV emulator about the expected authorization method. -The test case calls *SendInstallISO15118CertificateMethod* method in order to trigger the EV emulator to initiate installing a new certificate.	
	Memory State: *CertificateInstalled* from certificateType *V2GRootCertificate* *CertificateInstalled* from certificateType *MORootCertificate*	
	Reusable State(s): **State** is *EVConnectedPreSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The Charging Station sends a **Get15118EVCertificateRequest**	**2.** The OCTT responds with a **Get15118EVCertificateResponse** with **status** *Failed* **exiResponse** is omitted
		3. If an **AuthorizeRequest** is received, the testcase will FAIL and the OCTT reports why it failed.
Tool validations	* Step 1: Message: **Get15118EVCertificateRequest** - **action** must be *Install*	
	Post scenario validations: N/a	

Table 335. Test Case Id: TC_M_28_CS

Test case name	**Certificate Update EV - Success**	
Test case Id	TC_M_28_CS	
Use case Id(s)	M02	
Requirement(s)	M02.FR.01	
System under test	Charging Station	
Description	The EV initiates updating the existing certificate. The Charging Station forwards the update request to the CSMS.	
Purpose	To verify if the Charging Station is able to forward the request to the CSMS.	
Prerequisite(s)	- The Charging Station supports ISO 15118.	
Before (Preparations)	**Configuration State:** **ISO15118Ctrlr.ContractCertificateInstallationEnabled** is *true* -The test case calls *SendISO15118AuthorizationMethod* method with parameter *PnC* in order to inform the EV emulator about the expected authorization method. -The test case calls *SendInstallISO15118CertificateMethod* method in order to trigger the EV emulator to initiate installing a new certificate.	
	Memory State: *CertificateInstalled* from certificateType *V2GRootCertificate* *CertificateInstalled* from certificateType *MORootCertificate*	
	Reusable State(s): **State** is *EVConnectedPreSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The Charging Station sends a **Get15118EVCertificateRequest**	**2.** The OCTT responds with a **Get15118EVCertificateResponse** with **status** *Accepted* **exiResponse** *<Raw CertificateInstallationRes response for the EV, Base64 encoded.>*
	3. The Charging Station sends an **AuthorizeRequest**	**4.** The OCTT responds with an **AuthorizeResponse** with **status** *Accepted*
Tool validations	* Step 1: Message: **Get15118EVCertificateRequest** - **action** must be *Update*	
	Post scenario validations: N/a	

Table 336. Test Case Id: TC_M_29_CS

Test case name	**Certificate Update EV - Failed**	
Test case Id	TC_M_29_CS	
Use case Id(s)	M02	
Requirement(s)	M02.FR.01	
System under test	Charging Station	
Description	The EV initiates updating the existing certificate. The Charging Station forwards the update request to the CSMS.	
Purpose	To verify if the Charging Station is able to forward the request to the CSMS.	
Prerequisite(s)	- The Charging Station supports ISO 15118.	
Before (Preparations)	**Configuration State:** **ISO15118Ctrlr.ContractCertificateInstallationEnabled** is *true* -The test case calls *SendISO15118AuthorizationMethod* method with parameter *PnC* in order to inform the EV emulator about the expected authorization method. -The test case calls *SendInstallISO15118CertificateMethod* method in order to trigger the EV emulator to initiate installing a new certificate.	
	Memory State: *CertificateInstalled* from certificateType *V2GRootCertificate* *CertificateInstalled* from certificateType *MORootCertificate*	
	Reusable State(s): **State** is *EVConnectedPreSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The Charging Station sends a **Get15118EVCertificateRequest**	**2.** The OCTT responds with a **Get15118EVCertificateResponse** with **status** *Failed* **exiResponse** is omitted.
		3. If an **AuthorizeRequest** is received, the testcase will FAIL and the OCTT reports why it failed.
Tool validations	* Step 1: Message: **Get15118EVCertificateRequest** - **action** must be *Update*	
	Post scenario validations: N/a	

2.15. N Diagnostics

Table 337. Test Case Id: TC_N_01_CS

Test case name	**Get Monitoring Report - with monitoringCriteria**	
Test case Id	TC_N_01_CS	
Use case Id(s)	N02	
Requirement(s)	N02.FR.01, N02.FR.03,N02.FR.04, N02.FR.05, **N02.FR.06**, N02.FR.09, **N02.FR.12, N02.FR.13, N02.FR.14**	
System under test	Charging Station	
Description	CSMS requests a report of all monitors that match the given **monitoringCriteria**: Threshold, Delta or Periodic.	
Purpose	To test that Charging Station supports reporting of monitoring via **monitoringCriteria**. Starting with ThresholdMonitoring and then extending the set to check that combinations are handled properly.	
Prerequisite(s)	Charging Station has implemented device model monitoring and MonitoringCtrlr::Enabled = true.	
Before (Preparations)	**Configuration State:** The following monitors (on arbitrary variables) must be present as 'hard-wired' or 'preconfigured' or must have been configured by CSMS: - LowerThreshold - UpperThreshold - Delta - Periodic - PeriodicClockAligned	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. Charging Station responds with: **GetMonitoringReportResponse**	**1.** OCTT sends **GetMonitoringReportRequest** with: - **requestId** = *<Generated requestId>* - **monitoringCriteria** = { *ThresholdMonitoring* }
	3. Charging Station responds with: **NotifyMonitoringReportRequest**	**4.** OCTT sends **NotifyMonitoringReportResponse**
	Step 3 and 4 are repeated as often as needed to report all configuration variables.	
	6. Charging Station responds with: **GetMonitoringReportResponse**	**5.** OCTT sends **GetMonitoringReportRequest** with: - **requestId** = *<Generated requestId>* - **monitoringCriteria** = { *ThresholdMonitoring, DeltaMonitoring* }
	7. Charging Station responds with: **NotifyMonitoringReportRequest**	**8.** OCTT sends **NotifyMonitoringReportResponse**
	Step 7 and 8 are repeated as often as needed to report all configuration variables.	
	10. Charging Station responds with: **GetMonitoringReportResponse**	**9.** OCTT sends **GetMonitoringReportRequest** with: - **requestId** = *<Generated requestId>* - **monitoringCriteria** = { *DeltaMonitoring, PeriodicMonitoring* }
	11. Charging Station responds with: **NotifyMonitoringReportRequest**	**12.** OCTT sends **NotifyMonitoringReportResponse**
	Step 11 and 12 are repeated as often as needed to report all configuration variables.	

Test case name	**Get Monitoring Report - with monitoringCriteria**
Tool validations	* Step 2: Message: **GetMonitoringReportResponse** - **status** = *Accepted* - **statusInfo** is absent or **statusInfo.reasonCode** = *"NoError"* * Step 3: Message: **NotifyMonitoringReportRequest** - **requestId** = *<Generated requestId>* - **generatedAt** = *<timestamp at charging station>* - **seqNo** = *0* - **monitor.variableMonitoring.type** = *UpperThreshold* or *LowerThreshold* While **tbc** = *true* Message: **NotifyMonitoringReportRequest** - **seqNo** is incremented by 1 - **monitor.variableMonitoring.type** = *UpperThreshold* or *LowerThreshold* * Step 6: Message: **GetMonitoringReportResponse** - **status** = *Accepted* - **statusInfo** is absent or **statusInfo.reasonCode** = *"NoError"* * Step 7: Message: **NotifyMonitoringReportRequest** - **requestId** = *<Generated requestId>* - **generatedAt** = *<timestamp at charging station>* - **seqNo** = *0* - **monitor.variableMonitoring.type** = *UpperThreshold*, *LowerThreshold* or *Delta* While **tbc** = *true* Message: **NotifyMonitoringReportRequest** - **seqNo** is incremented by 1 - **monitor.variableMonitoring.type** = *UpperThreshold*, *LowerThreshold* or *Delta* * Step 10: Message: **GetMonitoringReportResponse** - **status** = *Accepted* - **statusInfo** is absent or **statusInfo.reasonCode** = *"NoError"* * Step 11: Message: **NotifyMonitoringReportRequest** - **requestId** = *<Generated requestId>* - **generatedAt** = *<timestamp at charging station>* - **seqNo** = *0* - **monitor.variableMonitoring.type** = *Delta*, *Periodic* or *PeriodicClockAligned* While **tbc** = *true* Message: **NotifyMonitoringReportRequest** - **seqNo** is incremented by 1 - **monitor.variableMonitoring.type** = *Delta*, *Periodic* or *PeriodicClockAligned*
	Post scenario validations: N/A

Table 338. Test Case Id: TC_N_02_CS

Test case name	**Get Monitoring Report - with component/variable**	
Test case Id	TC_N_02_CS	
Use case Id(s)	N02	
Requirement(s)	N02.FR.01, N02.FR.03,N02.FR.04, N02.FR.05, **N02.FR.08**, N02.FR.09	
System under test	Charging Station	
Description	CSMS requests a report of monitors that match the given list of components and variables.	
Purpose	To test that Charging Station supports reporting of monitoring via for a given list of components and optionally with variables.	
Prerequisite(s)	Charging Station has implemented device model monitoring and MonitoringCtrlr::Enabled = true.	
Before (Preparations)	**Configuration State:** N/a *Note: these are required variables for which a monitor can be expected to exist or it can be configured.*	
	Memory State: The following monitors must be present as 'hard-wired' or 'preconfigured' or must have been configured by CSMS: - Component "ChargingStation", variable "AvailabilityState", monitor type *Delta*" - Component "EVSE", *<Configured evseId>*, variable "AvailabilityState", monitor type *Delta*	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. Charging Station responds with: **GetMonitoringReportResponse**	**1.** OCTT sends **GetMonitoringReportRequest** with: - **requestId** = *<Generated requestId>* - **monitoringCriteria** is omitted - **componentVariable[0].component.name** = *"ChargingStation"* - **componentVariable[0].variable.name** = *"AvailabilityState"* - **componentVariable[1].component.name** = *"EVSE"* - **componentVariable[1].component.evse.id** = *<Configured evseId>* _Note: requesting AvailabilityState from ChargingStation and all monitors from Configured EVSE
	3. Charging Station responds with: **NotifyMonitoringReportRequest**	**4.** OCTT sends **NotifyMonitoringReportResponse**
	Step 3 and 4 are repeated as often as needed to report all configuration variables.	

Test case name	**Get Monitoring Report - with component/variable**	
Tool validations	* Step 2: Message: **GetMonitoringReportResponse** - **status** = *Accepted* - **statusInfo** is absent or **statusInfo.reasonCode** = *"NoError"*	
	* Step 3: Message: **NotifyMonitoringReportRequest** - **requestId** = *<Generated requestId>* - **generatedAt** = *<timestamp at charging station>* - **seqNo** = *0* - if **monitor.variable** = *"AvailabilityState"* then **monitor.variableMonitoring.type** = *Delta* *Note: fore EVSE #1 we request all monitors. There may be other monitors besides AvailabilityState.*	
	While **tbc** = *true*	Message: **NotifyMonitoringReportRequest** - **seqNo** is incremented by 1 - **monitor.variable** = *"AvailabilityState"* - **monitor.variableMonitoring.type** = *Delta* - **monitor.component_.name** = *ChargingStation* or *EVSE*
	Post scenario validations: Check that a monitor for AvailabilityState of type *Delta* is reported for both ChargingStation and COnfigured EVSE. If other monitors are present on Configured EVSE, then they will also be reported.	

Table 339. Test Case Id: TC_N_03_CS

Test case name	**Get Monitoring Report - with component criteria and component/variable**	
Test case Id	TC_N_03_CS	
Use case Id(s)	N02	
Requirement(s)	N02.FR.01, N02.FR.03,N02.FR.04, **N02.FR.05**, N02.FR.09, **N02.FR.10**, N02.FR.13	
System under test	Charging Station	
Description	CSMS requests a report of monitors that match both the component criteria and the given list of components and variables.	
Purpose	To test that Charging Station supports reporting of monitoring for both the component criteria and a given list of components and optionally with variables.	
Prerequisite(s)	Charging Station has implemented device model monitoring and MonitoringCtrlr::Enabled = true.	
Before (Preparations)	**Configuration State:** The following monitors must be present as 'hard-wired' or 'preconfigured' or must have been configured by CSMS: - Component "ChargingStation", variable "Power", monitor type *Periodic* - Component "EVSE", evse *<Configured evseId>*, variable "AvailabilityState", monitor type *Delta* *Note: these are required variables for which a monitor can be expected to exist or it can be configured.*	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. Charging Station responds with: **GetMonitoringReportResponse**	**1.** OCTT sends **GetMonitoringReportRequest** with: - **requestId** = *<Generated requestId1>* - **monitoringCriteria** is *ThresholdMonitoring* - **componentVariable[0].component.name** = *"ChargingStation"* - **componentVariable[0].variable.name** = *"AvailabilityState"* - **componentVariable[1].component.name** = *"EVSE"* - **componentVariable[1].component.evse.id** = *<Configured evseId>* - **componentVariable[1].variable.name** = *"AvailabilityState"* *Note: requesting AvailabilityState from ChargingStation and Configured EVSE, but filtered to _ThresholdMonitoring._*
	4. Charging Station responds with: **GetMonitoringReportResponse**	**3.** OCTT sends **GetMonitoringReportRequest** with: - **requestId** = *<Generated requestId2>* - **monitoringCriteria** is *DeltaMonitoring* - **componentVariable[0].component.name** = *"ChargingStation"* - **componentVariable[0].variable.name** = *"AvailabilityState"* - **componentVariable[1].component.name** = *"EVSE"* - **componentVariable[1].component.evse.id** = *<Configured evseId>* - **componentVariable[1].variable.name** = *"AvailabilityState"* *Note: requesting AvailabilityState from ChargingStation and Configured EVSE, but filtered to _Delta._*
	5. Charging Station responds with: **NotifyMonitoringReportRequest**	**6.** OCTT sends **NotifyMonitoringReportResponse**
	Step 5 and 6 are repeated as often as needed to report all configuration variables.	

Test case name	**Get Monitoring Report - with component criteria and component/variable**
Tool validations	* Step 2: Message: **GetMonitoringReportResponse** - **status** = *EmptyResultSet* - **statusInfo** is absent or **statusInfo.reasonCode** = *"NotFound"* * Step 4: Message: **GetMonitoringReportResponse** - **status** = *Accepted* - **statusInfo** is absent or **statusInfo.reasonCode** = *"NoError"* * Step 5: Message: **NotifyMonitoringReportRequest** - **requestId** = *<Generated requestId>* - **generatedAt** = *<timestamp at charging station>* - **seqNo** = *0* - **monitor.variableMonitoring.type** = *Delta* While **tbc** = *true* Message: **NotifyMonitoringReportRequest** - **seqNo** is incremented by 1 - **monitor.variableMonitoring.type** = *Delta*
	Post scenario validations: Check that nothing is reported for **requestId** = *<Generated requestId1>* and a monitor for AvailabilityState of type *Delta* is reported for both ChargingStation and EVSE #1 for **requestId** = *<Generated requestId2>*.

NOTE

Test Case Id: TC_N_04_CS

Since MonitoringCriterionEnum is defined as enumeration, this will most likely already be caught by the JSON parser.

Test case name	**Get Monitoring Report - for unknown component criteria**	
Test case Id	TC_N_04_CS	
Use case Id(s)	N02	
Requirement(s)	N02.FR.02	
System under test	Charging Station	
Description	CSMS sends a GetMonitoringReport with an invalid value in **monitoringCriteria**.	
Purpose	To test that Charging Station returns a *NotSupported* return code in response to an invalid value for **monitoringCriteria**.	
Prerequisite(s)	Charging Station has implemented device model monitoring and MonitoringCtrlr::Enabled = true.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Charging Station responds with: **GetMonitoringReportResponse**	**2.** OCTT sends **GetMonitoringReportRequest** with: - **requestId** = *<Generated requestId>* - **monitoringCriteria = {** ***DeltaMonitoring, <Configured Unsupported monitoringCriteria>*** **}** **- *componentVariable** is absent
Tool validations	* Step 1 Message: **GetMonitoringReportResponse** - **status** = *NotSupported* - **statusInfo** is absent or **statusInfo.reasonCode** = *"UnsupportedParam"* or **statusInfo.reasonCode** = *"InvalidValue"*	
	Post scenario validations: N/A	

Table 340. Test Case Id: TC_N_05_CS

Test case name	**Set Monitoring Base - success**	
Test case Id	TC_N_05_CS	
Use case Id(s)	N03	
Requirement(s)	N03.FR.01, N03.FR.03, N03.FR.04, N03.FR.05	
System under test	Charging Station	
Description	CSMS sends a SetMonitoringBaseRequest for All, FactoryDefault and HardWiredOnly.	
Purpose	To test that Charging Station supports all three monitoring base types.	
Prerequisite(s)	Charging Station has implemented device model monitoring and MonitoringCtrlr::Enabled = true.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. Charging Station responds with: **SetMonitoringBaseResponse**	**1.** OCTT sends **SetMonitoringBaseRequest** with: - **monitoringBase** = *All*
	4. Charging Station responds with: **SetMonitoringBaseResponse**	**3.** OCTT sends **SetMonitoringBaseRequest** with: - **monitoringBase** = *FactoryDefault*
	6. Charging Station responds with: **SetMonitoringBaseResponse**	**5.** OCTT sends **SetMonitoringBaseRequest** with: - **monitoringBase** = *HardWiredOnly*
Tool validations	* Step 2 Message: **SetMonitoringBaseResponse** - **status** = *Accepted* - **statusInfo** is absent or **statusInfo.reasonCode** = *"NoError"*	
	* Step 4 Message: **SetMonitoringBaseResponse** - **status** = *Accepted* - **statusInfo** is absent or **statusInfo.reasonCode** = *"NoError"*	
	* Step 6 Message: **SetMonitoringBaseResponse** - **status** = *Accepted* - **statusInfo** is absent or **statusInfo.reasonCode** = *"NoError"*	
	Post scenario validations: N/A	

Table 341. Test Case Id: TC_N_06_CS

Test case name	**Set Monitoring Base - test removal custom monitors**	
Test case Id	TC_N_06_CS	
Use case Id(s)	N03	
Requirement(s)	N03.FR.01, N03.FR.05	
System under test	Charging Station	
Description	CSMS sends a SetMonitoringBaseRequest for HardWiredOnly.	
Purpose	To test that Charging Station removes custom monitors when selecting a monitoring base, as specified explicitly in N03.FR.05 and less formally in the remark of the use case N03.	
Prerequisite(s)	Charging Station has implemented device model monitoring and MonitoringCtrlr::Enabled = true.	
Before (Preparations)	**Configuration State:** The following monitor must be present as 'preconfigured' or custom monitor configured by CSMS: - Component "ChargingStation", variable "AvailabilityState", monitor type *Delta* If it exists as a hardwired monitor, then the test will fail, because the test checks that it is removed when reverting back to only hardwired monitors. *Note: this is a required variable for which a monitor can be expected to exist or it can be configured.*	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	Check that monitor AvailabilityState exists.	
	2. Charging Station responds with: **GetMonitoringReportResponse**	**1.** OCTT sends **GetMonitoringReportRequest** with: - **requestId** = *<Generated requestId>* - **monitoringCriteria** is absent - **componentVariable[0].component.name** = *"ChargingStation"* - **componentVariable[0].variable.name** = *"AvailabilityState"*
	3. Charging Station responds with: **NotifyMonitoringReportRequest**	**4.** OCTT sends **NotifyMonitoringReportResponse**
	6. Charging Station responds with: **SetMonitoringBaseResponse**	**5.** OCTT sends **SetMonitoringBaseRequest** with: - **monitoringBase** = *HardWiredOnly*
	Check that monitor AvailabilityState has been removed.	
	8. Charging Station responds with: **GetMonitoringReportResponse**	**7.** OCTT sends **GetMonitoringReportRequest** with: - **requestId** = *<Generated requestId>* - **monitoringCriteria** is absent - **componentVariable[0].component.name** = *"ChargingStation"* - **componentVariable[0].variable.name** = *"AvailabilityState"*

Test case name	**Set Monitoring Base - test removal custom monitors**
Tool validations	* Step 2 Message: **GetMonitoringReportResponse** - **status** = *Accepted* - **statusInfo** is absent or **statusInfo.reasonCode** = *"NoError"*
	* Step 3: Message: **NotifyMonitoringReportRequest** - **requestId** = *<Generated requestId>* - **generatedAt** = *<timestamp at charging station>* - **seqNo** = *0* - **tbc** is absent or **tbc** = *false* - **monitor.variableMonitoring.type** = *Delta* - **monitor.component.name** = *"ChargingStation"* - **monitor.variable.name** = *"AvailabilityState"*
	* Step 6 Message: **SetMonitoringBaseResponse** - **status** = *Accepted* - **statusInfo** is absent or **statusInfo.reasonCode** = *"NoError"*
	* Step 8 Message: **GetMonitoringReportResponse** - **status** = *EmptyResultSet* - **statusInfo** is absent or **statusInfo.reasonCode** = *"NotFound"*

NOTE	*Test Case Id: TC_N_07_CS* Since MonitoringBaseEnumType is defined as enumeration, this will most likely already be caught by the JSON parser.

Test case name	**Set Monitoring Base - for unknown base type**	
Test case Id	TC_N_07_CS	
Use case Id(s)	N03	
Requirement(s)	N03.FR.02	
System under test	Charging Station	
Description	CSMS send a SetMonitoringBase with an invalid value in **monitoringBase**.	
Purpose	To test that Charging Station returns a *NotSupported* return code in response to an invalid value for **monitoringBase**.	
Prerequisite(s)	Charging Station has implemented device model monitoring and MonitoringCtrlr::Enabled = true.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. Charging Station responds with: **SetMonitoringBaseResponse**	**1.** OCTT sends **SetMonitoringBaseRequest** with: - **monitoringBase** = *<Configured unsupported_monitoringBase>*
Tool validations	* Step 2 Message: **SetMonitoringBaseResponse** - **status** = *NotSupported* - **statusInfo** is absent or **statusInfo.reasonCode** = *"UnsupportedParam"* or **statusInfo.reasonCode** = *"InvalidValue"*	
	Post scenario validations: N/A	

Table 342. Test Case Id: TC_N_08_CS

Test case name	**Set Variable Monitoring - one setMonitoringData element**	
Test case Id	TC_N_08_CS	
Use case Id(s)	N04	
Requirement(s)	N04.FR.01, N04.FR.02, N04.FR.08, N04.FR.11	
System under test	Charging Station	
Description	CSMS sends a request to activate a monitor on a single variable.	
Purpose	To test that Charging Station supports setting of a monitor on a variable.	
Prerequisite(s)	Charging Station has implemented device model monitoring and MonitoringCtrlr::Enabled = true.	
Before (Preparations)	**Configuration State:** This test case activates a monitor on the following variable: - Component "EVSE", evse "1", variable "AvailabilityState", monitor type *Delta* It assumes, that no monitor is active on this variable prior to the test. *Note: this is a required variable for which a monitor can be expected to exist or it can be configured.* *Note 2: Any other component/variable combination that supports monitoring could also be used for this test case.*	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. Charging Station responds with: **SetVariableMonitoringResponse**	*Install monitor* **1.** OCTT sends **SetVariableMonitoringRequest** with: - **setMonitoringData.value** = *1* - **setMonitoringData.type** = *Delta* - **setMonitoringData.severity** = *8* - **setMonitoringData.component.name** = *"EVSE"* - **setMonitoringData.component.evse.id** = *<Configured evseId>* - **setMonitoringData.variable.name** = *"AvailabilityState"*
	4. Charging Station responds with: **GetMonitoringReportResponse**	*Verify monitor is installed* **3.** OCTT sends **GetMonitoringReportRequest** with: - **requestId** = *<Generated requestId>* - **monitoringCriteria** is absent - **componentVariable[0].component.name** = *"EVSE"* - **componentVariable[0].component.evse.id** = *<Configured evseId>* - **componentVariable[0].variable.name** = *"AvailabilityState"*
	5. Charging Station responds with: **NotifyMonitoringReportRequest**	**6.** OCTT sends **NotifyMonitoringReportResponse**

Test case name	**Set Variable Monitoring - one setMonitoringData element**

Tool validations	* Step 2:
	Message: **SetVariableMonitoringResponse** with:
	setMonitoringResult = {
	- **status** = *Accepted*
	- **type** = *Delta*
	- **severity** = *8*
	- **component.name** = *"EVSE"*
	- **component.evse.id** = *<Configured evseId>*
	- **variable.name** = *"AvailabilityState"*
	- **statusInfo** is absent or **statusInfo.reasonCode** = *"NoError"*
	}

	* Step 4: Message: **GetMonitoringReportResponse** - **status** = *Accepted* - **statusInfo** is absent or **statusInfo.reasonCode** = *"NoError"*
	* Step 5: Message: **NotifyMonitoringReportRequest** - **requestId** = *<Generated requestId>* - **monitor.variableMonitoring.type** = *Delta* - **monitor.component.name** = *"EVSE"* - **monitor.component.evse.id** = *<Configured evseId>* - **monitor.variable.name** = *"AvailabilityState"*
	Post scenario validations:

Table 343. Test Case Id: TC_N_09_CS

Test case name	**Set Variable Monitoring - Multiple elements on different component and variable**	
Test case Id	TC_N_09_CS	
Use case Id(s)	N04	
Requirement(s)	N04.FR.01, N04.FR.02, N04.FR.08, N04.FR.11	
System under test	Charging Station	
Description	CSMS sends a request to activate monitors on different variables.	
Purpose	To test that Charging Station supports setting of multiple monitors on different variables.	
Prerequisite(s)	Charging Station has implemented device model monitoring and MonitoringCtrlr::Enabled = true.	
Before (Preparations)	**Configuration State:** This test case activates monitors on the following variables: - Component "EVSE", evse "1", variable "AvailabilityState", monitor type *Delta* - Component "ChargingStation", variable "AvailabilityState", monitor type *Delta* It assumes, that no monitor is active on these variables prior to the test. *Note: these are required variables for which a monitor can be expected to exist or it can be configured.* *Note 2: Any other component/variable combination that supports monitoring could also be used for this test case.*	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. Charging Station responds with: **SetVariableMonitoringResponse**	*Install monitors* **1.** OCTT sends **SetVariableMonitoringRequest** with: - **setMonitoringData[0].value** = *1* - **setMonitoringData[0].type** = *Delta* - **setMonitoringData[0].severity** = *<Configured severity>* - **setMonitoringData[0].component.name** = *"EVSE"* - **setMonitoringData[0].component.evse.id** = *<Configured evseId>* - **setMonitoringData[0].variable.name** = *"AvailabilityState"* - **setMonitoringData[1].value** = *1* - **setMonitoringData[1].type** = *Delta* - **setMonitoringData[1].severity** = *<Configured severity>* - **setMonitoringData[1].component.name** = *"ChargingStation"* - **setMonitoringData[1].variable.name** = *"AvailabilityState"*

Test case name	Set Variable Monitoring - Multiple elements on different component and variable	
	4. Charging Station responds with: **GetMonitoringReportResponse**	*Verify monitors are installed* **3.** OCTT sends **GetMonitoringReportRequest** with: - **requestId** = *<Generated requestId>* - **monitoringCriteria** is absent - **componentVariable[0].component.name** = *"EVSE"* - **componentVariable[0].component.evse.id** = *1* - **componentVariable[0].variable.name** = *"AvailabilityState"* - **componentVariable[1].component.name** = *"ChargingStation"* - **componentVariable[1].variable.name** = *"AvailabilityState"*
	5. Charging Station responds with: **NotifyMonitoringReportRequest**	**6.** OCTT sends **NotifyMonitoringReportResponse**
	Step 5 and 6 may be repeated if the result is not sent in one report message.	
Tool validations	* Step 2: Message: **SetVariableMonitoringResponse** with: **setMonitoringResult**[1] - **id** = *<id of new monitor>* - **status** = *Accepted* - **type** = *Delta* - **severity** = *8* - **component.name** = *"EVSE"* - **component.evse.id** = *<Configured evseId>* - **variable.name** = *"AvailabilityState"* - **statusInfo** is absent or **statusInfo.reasonCode** = *"NoError"* **setMonitoringResult**[2] - **id** = *<id of new monitor>* - **status** = *Accepted* - **type** = *Delta* - **severity** = *8* - **component.name** = *"ChargingStation"* - **variable.name** = *"AvailabilityState"* - **statusInfo** is absent or **statusInfo.reasonCode** = *"NoError"* * Step 4: Message: **GetMonitoringReportResponse** - **status** = *Accepted* - **statusInfo** is absent or **statusInfo.reasonCode** = *"NoError"* * Step 5: Message: **NotifyMonitoringReportRequest** - **requestId** = *<Generated requestId>* - **generatedAt** = *<timestamp at charging station>* - **seqNo** = *0*	
	while **tbc** is true	Expect **NotifyMonitoringReportRequest** - **seqNo** is incremented by 1

Table 344. Test Case Id: TC_N_10_CS

Test case name	**Set Variable Monitoring - Multiple monitors on the same component and variable**	
Test case Id	TC_N_10_CS	
Use case Id(s)	N04	
Requirement(s)	N04.FR.01, N04.FR.02, N04.FR.08, N04.FR.11	
System under test	Charging Station	
Description	CSMS sets multiple monitors on the same component/variable combination.	
Purpose	To test that Charging Station supports multiple monitors on same component/variable combination.	
Prerequisite(s)	Charging Station has implemented device model monitoring and MonitoringCtrlr::Enabled = true.	
Before (Preparations)	**Configuration State:** This test case activates two monitors on the following variable: - Component "EVSE", evse "1", variable "AvailabilityState", monitor type *Delta* *Note: it does not make any practical sense to install two _Delta* monitors on same variable with different severity, because a *Delta* monitor on a non-numeric variable is triggered by any change. However, the specification allows for it, therefore we use this variable, because it must exist. If the variable "Power" can be monitored on an EVSE, then it is much more realistic to use that with a combination of two different UpperThresholds and severities._	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. Charging Station responds with: **SetVariableMonitoringResponse**	*Install monitors* **1.** OCTT sends **SetVariableMonitoringRequest** with: - **setMonitoringData[0].value** = *1* - **setMonitoringData[0].type** = *Delta* - **setMonitoringData[0].severity** = *8* - **setMonitoringData[0].component.name** = *"EVSE"* - **setMonitoringData[0].component.evse.id** = *<Configured evseId>* - **setMonitoringData[0].variable.name** = *"AvailabilityState"* - **setMonitoringData[1].value** = *1* - **setMonitoringData[1].type** = *Delta* - **setMonitoringData[1].severity** = *7* - **setMonitoringData[1].component.name** = *"EVSE"* - **setMonitoringData[1].component.evse.id** = *<Configured evseId>* - **setMonitoringData[1].variable.name** = *"AvailabilityState"*

Test case name	**Set Variable Monitoring - Multiple monitors on the same component and variable**

Tool validations	* Step 2:
	Message: **SetVariableMonitoringResponse** with (in arbitrary order):
	setMonitoringResult[1] = {
	- **id** = *<id of new monitor>*
	- **status** = *Accepted*
	- **type** = *Delta*
	- **severity** = *8*
	- **component.name** = *"EVSE"*
	- **component.evse.id** = *<Configured evseId>*
	- **variable.name** = *"AvailabilityState"*
	- **statusInfo** is absent or **statusInfo.reasonCode** = *"NoError"*
	}
	setMonitoringResult[2] = {
	- **id** = *<id of new monitor>*
	- **status** = *Accepted*
	- **type** = *Delta*
	- **severity** = *7*
	- **component.name** = *"EVSE"*
	- **component.evse.id** = *<Configured evseId>*
	- **variable.name** = *"AvailabilityState"*
	- **statusInfo** is absent or **statusInfo.reasonCode** = *"NoError"*
	}

	Post scenario validations: N/A

Table 345. Test Case Id: TC_N_11_CS

Test case name	**Set Variable Monitoring - Unknown component**	
Test case Id	TC_N_11_CS	
Use case Id(s)	N04	
Requirement(s)	N04.FR.03	
System under test	Charging Station	
Description	CSMS tries to set a monitor on an unknown component.	
Purpose	To test that Charging Station checks whether a component exists.	
Prerequisite(s)	Charging Station has implemented device model monitoring and MonitoringCtrlr::Enabled = true.	
Before (Preparations)	**Configuration State:** This test case activates a monitor on an existing component on non-existing **evse** and then on a non-existing component "NonExistent": - Component "EVSE", evse "99", variable "AvailabilityState", monitor type *Delta* - Component "NonExistent", variable "Power", monitor type *UpperThreshold* *Note: this assumes, that EVSE #99 does not exist.* *The response to the "NonExistent" component can be either UnknownComponent or UnknownVariable, because both will not exist.*	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**

	2. Charging Station responds with:
	SetVariableMonitoringResponse
		Install monitors
		1. OCTT sends **SetVariableMonitoringRequest** with:
		setMonitoringData[1] = {
		- **setMonitoringData[0].value** = *1*
		- **setMonitoringData[0].type** = *Delta*
		- **setMonitoringData[0].severity** = *<Configured severity>*
		- **setMonitoringData[0].component.name** = *"EVSE"*
		- **setMonitoringData[0].component.evse.id** = *99*
		- **setMonitoringData[0].variable.name** = *"AvailabilityState"*
		}
		setMonitoringData[2] = {
		- **setMonitoringData[1].value** = *1234.0*
		- **setMonitoringData[1].type** = *UpperThreshold*
		- **setMonitoringData[1].severity** = *<Configured severity>*
		- **setMonitoringData[1].component.name** = *"NonExistent"*
		- **setMonitoringData[1].variable.name** = *"Power"*
		}

Test case name	**Set Variable Monitoring - Unknown component**
Tool validations	* Step 2: Message: **SetVariableMonitoringResponse** with (in arbitrary order): - **id** is absent - **status** = *UnknownComponent* or *Rejected* - **type** = *Delta* - **severity** = *<Configured severity>* - **component.name** = *"EVSE"* - **component.evse.id** = *99* - **variable.name** = *"AvailabilityState"* - **statusInfo** is absent or **statusInfo.reasonCode** = *"UnknownEVSE"* or **statusInfo** = *"NotFound"* - **id** is absent - **status** = *UnknownComponent* (*UnknownVariable* will also be allowed, but is less accurate) - **type** = *UpperThreshold* - **severity** = *<Configured severity>* - **component.name** = *"NonExistent"* - **variable.name** = *"AvailabilityState"* - **statusInfo** is absent or **statusInfo.reasonCode** = *"NotFound"*
	Post scenario validations: N/A

Table 346. Test Case Id: TC_N_12_CS

Test case name	**Set Variable Monitoring - Value out of range - Delta monitor**	
Test case Id	TC_N_12_CS	
Use case Id(s)	N04	
Requirement(s)	N04.FR.14	
System under test	Charging Station	
Description	CSMS tries to set a delta monitor with a value that is out of range.	
Purpose	To test that Charging Station checks that value is within range of variable.	
Prerequisite(s)	Charging Station has implemented device model monitoring and MonitoringCtrlr::Enabled = true. This test case assumes the following component exists and can be monitored: - Component "EVSE", evse "1", variable "AvailabilityState", monitor type *Delta* *Note: Variable _AvailabilityState* is mandatory for an EVSE and it is likely (but not guaranteed), that it can be monitored._	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. Charging Station responds with: **SetVariableMonitoringResponse**	*Install monitors* **1.** OCTT sends **SetVariableMonitoringRequest** with: - **setMonitoringData[0].value** = *-1* - **setMonitoringData[0].type** = *Delta* - **setMonitoringData[0].severity** = *<Configured severity>* - **setMonitoringData[0].component.name** = *"EVSE"* - **setMonitoringData[0].component.evse.id** = *<Configured evseId>* - **setMonitoringData[0].variable.name** = *"AvailabilityState"*

Tool validations	* Step 2:
	Message: **SetVariableMonitoringResponse** with (in arbitrary order):
	setMonitoringResult = {
	- **id** is absent
	- **status** = *Rejected*
	- **type** = *Delta*
	- **severity** = *<Configured severity>*
	- **component.name** = *"EVSE"*
	- **component.evse.id** = *<Configured evseId>*
	- **variable.name** = *"AvailabilityState"*
	- **statusInfo** is absent or **statusInfo.reasonCode** = *"ValueOutOfRange"* or **statusInfo.reasonCode** = "ValuePositiveOnly"
	}	

	Post scenario validations: N/A	

Table 347. Test Case Id: TC_N_13_CS

Test case name	**Set Variable Monitoring - Value out of range - Threshold monitor**	
Test case Id	TC_N_13_CS	
Use case Id(s)	N04	
Requirement(s)	N04.FR.13	
System under test	Charging Station	
Description	CSMS tries to set a threshold monitor with a value that is out of range.	
Purpose	To test that Charging Station checks that value is within range of variable.	
Prerequisite(s)	Charging Station has implemented device model monitoring and MonitoringCtrlr::Enabled = true. This test case assumes the *<Configured threshold monitor component variable>* component.variable exists and can be monitored and has **variableCharacteristics.maxLimit** < *<Configured threshold monitor value>* + *Note: Variable _Power(maxLimit)* is mandatory for an EVSE, but the actual value not, but it is likely (but not guaranteed), that it can be monitored._	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. Charging Station responds with: **SetVariableMonitoringResponse**	*Install monitors* **1.** OCTT sends **SetVariableMonitoringRequest** with: - **setMonitoringData[0].value** = *<Configured threshold monitor value>* - **setMonitoringData[0].type** = *UpperThreshold* - **setMonitoringData[0].severity** = *<Configured severity>* - **setMonitoringData[0].component.name** = *<Configured threshold monitor component variable>* - **setMonitoringData[0].component.evse.id** = *<Configured evseId>* - **setMonitoringData[0].variable.name** = *<Configured threshold monitor component variable>*

Tool validations	* Step 2:
	Message: **SetVariableMonitoringResponse** with (in arbitrary order):
	setMonitoringResult = {
	- **id** is absent
	- **status** = *Rejected*
	- **type** = *UpperThreshold*
	- **severity** = *<Configured severity>*
	- **component.name** = *<Configured threshold monitor component variable>*
	- **component.evse.id** = *<Configured evseId>*
	- **variable.name** = *<Configured threshold monitor component variable>*
	- **statusInfo** is absent or **statusInfo.reasonCode** = *"ValueOutOfRange"*
	}	

	Post scenario validations: N/A	

Table 348. Test Case Id: TC_N_15_CS

Test case name	**Set Variable Monitoring - Duplicate Variable type/severity combination**	
Test case Id	TC_N_15_CS	
Use case Id(s)	N04	
Requirement(s)	N04.FR.10	
System under test	Charging Station	
Description	CSMS sets multiple monitors on the same component/variable combination with same severity and type.	
Purpose	To test that Charging Station rejects multiple monitors on same component/variable combination when having the same severity and type.	
Prerequisite(s)	Charging Station has implemented device model monitoring and MonitoringCtrlr::Enabled = true.	
Before (Preparations)	**Configuration State:** This test case activates two monitors on the following variable: - Component "EVSE", evse "1", variable "AvailabilityState", monitor type *Delta* + *Note: it does not make any practical sense to install two _Delta* monitors on same variable with different severity, because a *Delta* monitor on a non-numeric variable is triggered by any change. However, the specification allows for it, therefore we use this variable, because it must exist. If the variable "Power" can be monitored on an EVSE, then it is much more realistic to use that with a combination of two different UpperThresholds and severities._	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. Charging Station responds with: **SetVariableMonitoringResponse**	*Install monitors with same severity and of type _Delta* **1.** OCTT sends **SetVariableMonitoringRequest** with: - **setMonitoringData[0].value** = *1* - **setMonitoringData[0].type** = *Delta* - **setMonitoringData[0].severity** = *<Configured severity>* - **setMonitoringData[0].component.name** = *"EVSE"* - **setMonitoringData[0].component.evse.id** = *<Configured evseId>* - **setMonitoringData[0].variable.name** = *"AvailabilityState"* - **setMonitoringData[1].value** = *1* - **setMonitoringData[1].type** = *Delta* - **setMonitoringData[1].severity** = *<Configured severity>* - **setMonitoringData[1].component.name** = *"EVSE"* - **setMonitoringData[1].component.evse.id** = *<Configured evseId>* - **setMonitoringData[1].variable.name** = *"AvailabilityState"*

Test case name	**Set Variable Monitoring - Duplicate Variable type/severity combination**

Tool validations	* Step 2:
	Message: **SetVariableMonitoringResponse** with (in arbitrary order):
	setMonitoringResult[1] = {
	- **id** = *<id of new monitor>*
	- **status** = *Accepted*
	- **type** = *Delta*
	- **severity** = *<Configured severity>*
	- **component.name** = *"EVSE"*
	- **component.evse.id** = *<Configured evseId>*
	- **variable.name** = *"AvailabilityState"*
	- **statusInfo** is absent or **statusInfo.reasonCode** = *"NoError"*
	}
	setMonitoringResult[2] = {
	- **status** = *Duplicate*
	- **type** = *Delta*
	- **severity** = *<Configured severity>*
	- **component.name** = *"EVSE"*
	- **component.evse.id** = *<Configured evseId>*
	- **variable.name** = *"AvailabilityState"*
	- **statusInfo** is absent or **statusInfo.reasonCode** = *"InvalidValue"*
	}

	Post scenario validations: N/A

Table 349. Test Case Id: TC_N_16_CS

Test case name	**Set Monitoring Level - Success**	
Test case Id	TC_N_16_CS	
Use case Id(s)	N05	
Requirement(s)	N05.FR.01, N05.FR.03	
System under test	Charging Station	
Description	CSMS sets a monitoring level after which only monitors with lower or equal level are reported.	
Purpose	To test that Charging Station accepts monitoring message and correctly filters events.	
Prerequisite(s)	Charging Station has implemented device model monitoring and MonitoringCtrlr::Enabled = true.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: This test case activates a monitor on the following variable: - Component "EVSE", variable "AvailabilityState", monitor type *Delta*, severity 8 It assumes that no monitor is active on this variable at start of the test.	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	Set a monitoring level that suppresses the notification	
	2. Charging Station responds with: **SetMonitoringLevelResponse**	**1.** OCTT sends: **SetMonitoringLevelRequest** with: **severity** = *7*
	4 Charging Station does NOT send **NotifyEventRequest** for configured EVSE	**3.** *Plugin cable to configured EVSE to make it* _*Occupied* and test that notification is suppressed._
Tool validations	* Step 2: Message: **SetMonitoringLevelResponse** with: **status** = *Accepted* **statusInfo** is absent or **statusInfo.reasonCode** = *"NoError"*	
	Post scenario validations: Verify that no event notification is sent for the configured EVSE.	

Table 350. Test Case Id: TC_N_17_CS

Test case name	**Set Monitoring Level - Out of range**	
Test case Id	TC_N_17_CS	
Use case Id(s)	N05	
Requirement(s)	N05.FR.02	
System under test	Charging Station	
Description	CSMS sets a monitoring level with an out of range value.	
Purpose	To test that Charging Station rejects monitoring message with out of range severity.	
Prerequisite(s)	Charging Station has implemented device model monitoring and MonitoringCtrlr::Enabled = true.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**2.** Charging Station responds with: **SetMonitoringLevelResponse**	**1.** OCTT sends: **SetMonitoringLevelRequest** with: **severity** = *10*
	4. Charging Station responds with: **SetMonitoringLevelResponse**	**3.** OCTT sends: **SetMonitoringLevelRequest** with: **severity** = *-1*
Tool validations	* Step 2: Message: **SetMonitoringLevelResponse** with: - **status** = *Rejected* - **statusInfo** is absent or **statusInfo.reasonCode** = *"ValueOutOfRange"* or **statusInfo.reasonCode** = *"ValueTooHigh"*	
	* Step 4: Message: **SetMonitoringLevelResponse** with: - **status** = *Rejected* - **statusInfo** is absent or **statusInfo.reasonCode** = *"ValueOutOfRange"* or **statusInfo.reasonCode** = *"ValueTooLow"*	
	Post scenario validations: N/A	

Table 351. Test Case Id: TC_N_18_CS

Test case name	**Clear Monitoring - Success**	
Test case Id	TC_N_18_CS	
Use case Id(s)	N06	
Requirement(s)	N06.FR.01, N06.FR.05	
System under test	Charging Station	
Description	CSMS clears a monitor that is identified by its **id**.	
Purpose	To test that Charging Station clears the monitor.	
Prerequisite(s)	Charging Station has implemented device model monitoring and MonitoringCtrlr::Enabled = true.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: - Component "ChargingStation", variable "AvailabilityState" - Component "EVSE", variable "AvailabilityState"	
	Reusable State(s): N/a	
Main (Test scenario)	**2.** Charging Station responds with: **ClearVariableMonitoringResponse**	**1.** OCTT sends: **ClearVariableMonitoringRequest** with: - **id** = { ID1, ID2 }
	4. Charging Station responds with: **GetMonitoringReportResponse**	*Verify monitors are cleared* **3.** OCTT sends **GetMonitoringReportRequest** with: - **requestId** = *<Generated requestId>* - **monitoringCriteria** is absent - **componentVariable[0].component.name** = *"ChargingStation"* - **componentVariable[0].variable.name** = *"AvailabilityState"* - **componentVariable[1].component.name** = *"EVSE"* - **componentVariable[1].component.evse.id** = *1* - **componentVariable[1].variable.name** = *"AvailabilityState"*
Tool validations	* Step 2: Message: **ClearVariableMonitoringResponse** with (in arbitrary order): **clearMonitoringResult**[1]: - **status** = *Accepted* - **id** = <ID1> - **statusInfo** is absent or **statusInfo.reasonCode** = *"NoError"* **clearMonitoringResult**[2]: - **status** = *Accepted* - **id** = <ID2> - **statusInfo** is absent or **statusInfo.reasonCode** = *"NoError"*	
	* Step 4: Message: **GetMonitoringReportResponse** with: - **status** = *EmptyResultSet* - **statusInfo** is absent or **statusInfo.reasonCode** = *"NotFound"*	
	Post scenario validations: N/A	

Table 352. Test Case Id: TC_N_19_CS

Test case name	**Clear Monitoring - Not found**	
Test case Id	TC_N_19_CS	
Use case Id(s)	N06	
Requirement(s)	N06.FR.02	
System under test	Charging Station	
Description	CSMS clears a monitor that does not exist.	
Purpose	To test that Charging Station responds with *NotFound* result.	
Prerequisite(s)	Charging Station has implemented device model monitoring and MonitoringCtrlr::Enabled = true.	
Before (Preparations)	**Configuration State:** OCTT Sends a GetMonitoringReportRequest, the CS then reports all existsing monitors if it has any. If any monitors exist the tool will take the highest id number and add 1, if no monitors are reported a preconfigured number is used.	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	This test uses a monitor id, that is expected not to exist.	
	2. Charging Station responds with: **ClearVariableMonitoringResponse**	**1.** OCTT sends: **ClearVariableMonitoringRequest** with: - **id** *monitor id from the Preparations*
Tool validations	* Step 2: Message: **ClearVariableMonitoringResponse** with: **clearMonitoringResult**: - **status** = *NotFound* - **id** = *123456* - **statusInfo** is absent or **statusInfo.reasonCode** = *"NotFound"*	
	Post scenario validations: N/A	

Table 353. Test Case Id: TC_N_20_CS

Test case name	**Alert Event - Threshold value exceeded**
Test case Id	TC_N_20_CS
Use case Id(s)	N07
Requirement(s)	N07.FR.06, N07.FR.07, N07.FR.16, N07.FR.17
System under test	Charging Station
Description	A monitored variable exceeds a threshold monitor and causes a **NotifyEventRequest** message to be sent.
Purpose	To test that Charging Station supports threshold monitors
Prerequisite(s)	Charging Station has implemented device model monitoring and MonitoringCtrlr::Enabled = true.
Before (Preparations)	**Configuration State:** N/a
	Memory State: This test requires the Monitoring Base to be set to *All*. - **SetMonitoringBaseRequest** with **monitoringBase** = *All*. Futhermore this test requires the existence of a *LowerThreshold* and *UpperThreshold* monitor on a (numerical) variable. Since it is not mandated which variables are required to be monitored, this test used the variable "Power" of component "EVSE". - **setMonitoringData[0].value** = *<Configured threshold monitor value>* - **setMonitoringData[0].type** = *UpperThreshold* - **setMonitoringData[0].severity** = *<Configured severity>* - **setMonitoringData[0].component.name** = *<Configured threshold monitor component variable>* - **setMonitoringData[0].component.evse.id** = *<Configured evseId>* - **setMonitoringData[0].variable.name** = *<Configured threshold monitor component variable>* Notes: *- If componentVariable is set to "Power" or "Current", the value is set to the configured maxLimit 100.0* *- Take a threshold that can easily be exceeded.*
	Reusable State(s): N/a

Test case name	**Alert Event - Threshold value exceeded**	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *EnergyTransferStarted* or manually trigger the monitor. Notes: *If componentVariable is set to "Power" or "Current" EnergyTransferStarted will trigger the monitor. If another componentvariable is chosen a manual action is needed to trigger the monitor.*	
	2. Charging Station sends a **NotifyEventRequest** with: - Power exceeding upper threshold	**3.** OCTT responds with a **NotifyEventResponse**
	5. Charging Station responds with a **SetVariableMonitoringResponse** with: - **status** *Accepted*	**4.** OCTT sends a **SetVariableMonitoringRequest** with: - **type** *LowerThreshold* - **component.name** *<Configured threshold monitor component variable>* - **component.evse.id** *<configured evseId>* - **variable.name** *<Configured threshold monitor component variable>* - **value** *<Configured threshold monitor2 value>* Notes: *- If componentVariable is set to "Power" or "Current", the value is set to the configured maxLimit 10.0* *- Take a threshold that won't be exceeded.*
	6. Execute **Reusable State** *StopAuthorized* or manually trigger the second monitor. Notes: *If componentVariable is set to "Power" or "Current" EnergyTransferStarted will trigger the monitor. If another componentvariable is chosen a manual action is needed to trigger the monitor.*	
	7. Charging Station sends: **NotifyEventRequest** for 2 events: - Returning below upper threshold (*cleared*) - Dropping below lower threshold	**8.** OCTT responds: **NotifyEventResponse**
	Notes: *Steps 2, 3, 7, and 8 may be repeated if the data is sent using two requests instead of one. Depending on the configuration the Charging Station may also send other notifications during step 4 and 9.*	
Tool validations	* Step 2: Message: **NotifyEventRequest** with: - **generatedAt** = <time of generation at Charging Station> - **seqNo** = *0* and an **eventData** element with: - **eventId** = <id1> - **timestamp** = <time of event at Charging Station> - **trigger** = *Alerting* - **actualValue** = <current power> (must be **>** *<Configured threshold monitor value>*) - **cleared** is absent or **cleared** = *false* - **transactionId** = <transaction id> (delivery of power is always in transaction) - **variableMonitoringId** = <monitor id1> - **component.name** = *<Configured threshold monitor component variable>* - **component.evse.id** = *<Configured evseId>* - **variable.name** = *<Configured threshold monitor component variable>* *Other **eventData** elements can be ignored.*	

Test case name	**Alert Event - Threshold value exceeded**
	* Step 7: Message: **NotifyEventRequest** with: - **generatedAt** = <time of generation at Charging Station> - **seqNo** = *0* and an **eventData** element with: - **eventId** = <id2> - **timestamp** = <time of event at Charging Station> - **trigger** = *Alerting* - **actualValue** = <current power> (must be **=<** *<Configured threshold monitor value>*) - **cleared** is true - **transactionId** = <transaction id> (delivery of power is always in transaction) - **variableMonitoringId** = <monitor id1> - **eventNotificationType** = *CustomMonitor* - **component.name** = *<Configured threshold monitor component variable>* - **component.evse.id** = *<Configured evseId>* - **variable.name** = *<Configured threshold monitor component variable>* and an **eventData** element with: - **eventId** = <id3> - **timestamp** = <time of event at Charging Station> - **trigger** = *Alerting* - **actualValue** = <current power> (must be **<** *<Configured threshold monitor2 value>*) - **cleared** is absent or **cleared** is false - **transactionId** = <transaction id> (delivery of power is always in transaction) - **variableMonitoringId** = <monitor id2> - **eventNotificationType** = *CustomMonitor* - **component.name** = *<Configured threshold monitor component variable>* - **component.evse.id** = *<Configured evseId>* - **variable.name** = *<Configured threshold monitor component variable>* *Other **eventData** elements can be ignored. This can also be sent in two NotifyEventRequests, instead of one.*
	Post scenario validations: N/A

Table 354. Test Case Id: TC_N_21_CS

Test case name	**Alert Event - Caused by hardwired trigger**	
Test case Id	TC_N_21_CS	
Use case Id(s)	N07	
Requirement(s)		
System under test	Charging Station	
Description	An event that is hardwired in the firmware is reported.	
Purpose	To test that Charging Station reports this as a *HardWiredNotification*.	
Prerequisite(s)	Charging Station has implemented device model monitoring and MonitoringCtrlr::Enabled = true. This test assumes the existence of a hardwired notification in the Charging Station. The OCPP specification does not mandate any hardwired notifications, so it is up to the tester to select a certain notification and cause it to trigger the sending of an NotifyEventRequest.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	Tester triggers Charging Station to send a hardwired notification for component _X and variable *Y._*	
	1. Charging Station sends: **NotifyEventRequest**	**2.** OCTT responds: **NotifyEventResponse**
Tool validations	* Step 1: Message: **NotifyEventRequest** with: - **generatedAt** = <time of generation at Charging Station> - **seqNo** = *0* and an **eventData** element with: - **eventNotificationType** = *HardWiredNotification* *Other **eventData** elements are not relevant for this test.*	
	Post scenario validations: N/A	

Table 355. Test Case Id: TC_N_22_CS

Test case name	**Offline Notification - OfflineMonitoringEventQueuingSeverity set equal or lower than severityLevel of the monitor**	
Test case Id	TC_N_22_CS	
Use case Id(s)	N07	
Requirement(s)	N07.FR.04	
System under test	Charging Station	
Description	Charging Station queues event notifications when offline.	
Purpose	To test that Charging Station will queue event notifications with a severity equal or lower than `OfflineMonitoringEventQueuingSeverity`.	
Prerequisite(s)	Charging Station is online at start of test for configuration. CS has implemented device model monitoring and MonitoringCtrlr::Enabled = true.	
Before (Preparations)	**Configuration State:** SetConfiguration with: - **component.name** = *"MonitoringCtrlr"* - **variable.name** = *"OfflineQueuingSeverity"* - **attributeValue** = *<Configured Severity>*	
	Memory State: Charging Station has a custom or predefined monitor on AvailabilityState for Configured EVSE with *severity = <Configured severity>*	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	Take Charging Station offline.	
	2. *Charging Station queues event notification for EVSE #1::_AvailabilityState._*	**1.** *Plug a cable into EVSE #1 to generate an event notification for _AvailabilityState._*
	Note(s): *The tool will now wait for <Configured Transaction Duration> seconds*	
	Manual Action: *Bring Charging Station back online.*	
	3. Charging Station sends **NotifyEventRequest**	**4.** OCTT responds with **NotifyEventResponse**
	Steps 3 and 4 repeat for all queued events during the offline period	
Tool validations	* Step 1: *no communication*	
	* Step 3: Validate that the following **NotifyEventRequest** message was received: with an **eventData** element with: - **eventData[0].trigger** = *Delta* - **eventData[0].actualValue** = *"Occupied"* - **eventData[0].component.name** = *"EVSE"* - **eventData[0].component.evse.id** = *<Configured evseId>* - **eventData[0].variable.name** = *"AvailabilityState"*	
	Post scenario validations: N/a	

Table 356. Test Case Id: TC_N_23_CS

Test case name	**Offline Notification - OfflineMonitoringEventQueuingSeverity set higher than severityLevel of the monitor**	
Test case Id	TC_N_23_CS	
Use case Id(s)	N07	
Requirement(s)	N07.FR.04s	
System under test	Charging Station	
Description	Charging Station does not queue event notifications when offline.	
Purpose	To test that Charging Station does not queue event notifications with a severity higher than `OfflineMonitoringEventQueuingSeverity`.	
Prerequisite(s)	Charging Station is online at start of test for configuration. CS has implemented device model monitoring and MonitoringCtrlr::Enabled = true.	
Before (Preparations)	**Configuration State:** SetConfiguration with: - **component.name** = *"MonitoringCtrlr"* - **variable.name** = *"OfflineQueuingSeverity"* - **attributeValue** = *<Configured Severity>*	
	Memory State: Charging Station has a custom or predefined monitor on AvailabilityState for Configured EVSE with *severity = <Configured severity> + 1*	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	<u>Manual Action</u>: *Connect the EV and EVSE.*	
	1. The Charging Station notifies the CSMS about the status change of the connector.	**2.** The OCTT responds accordingly.
	<u>Note(s)</u>: *Step 3, 4, 5, 6, 7, and 8 need to be executed when* ***TxStartPoint*** *contains EVConnected OR ParkingBayOccupancy*	
	3. The Charging Station sends a **TransactionEventRequest**	**4.** The OCTT responds with a **TransactionEventResponse**
	<u>Manual Action</u>: *Take Charging Station offline.*	
	<u>Manual Action</u>: *Disconnect the EV and EVSE.*	
	<u>Manual Action</u>: *Connect the EV and EVSE.*	
	<u>Note(s)</u>: *The tool will now wait for <Configured Transaction Duration> seconds*	
	<u>Manual Action</u>: *Bring Charging Station back online.*	
	5. The Charging Station sends a **TransactionEventRequest**	**6.** The OCTT responds with a **TransactionEventResponse**
	7. The Charging Station sends a **TransactionEventRequest**	**8.** The OCTT responds with a **TransactionEventResponse**
	<u>Note(s)</u>: *The CS should not send a StatusNotificationRequest or NotifyEventRequest*	

Test case name	**Offline Notification - OfflineMonitoringEventQueuingSeverity set higher than severityLevel of the monitor**
Tool validations	* Step 1: Message: **StatusNotificationRequest** - **evseId** *<configured evseId>* - **connectorId** *<configured connectorId>* - **connectorStatus** must be *Occupied* Message: **NotifyEventRequest** - **eventData[0].trigger** must be *Delta* - **eventData[0].actualValue** must be *Occupied* - **eventData[0].component.name** must be *Connector* - **eventData[0].variable.name** must be *AvailabilityState* - **evse.id** *<configured evseId>* - **connector.id** *<configured connectorId>*
	* Step 3: Message: **TransactionEventRequest** - **triggerReason** must be *CablePluggedIn* - **transactionInfo.chargingState** must be *EVConnected*

Table 357. Test Case Id: TC_N_24_CS

Test case name	**Set Variable Monitoring - Periodic event**	
Test case Id	TC_N_24_CS	
Use case Id(s)	N07, N08	
Requirement(s)	N07.FR.20, N08.FR.01, N08.FR.05	
System under test	Charging Station	
Description	Charging Station sends a periodic event .	
Purpose	To test that Charging Station sends periodic events	
Prerequisite(s)	Charging Station has implemented device model monitoring and MonitoringCtrlr::Enabled = true.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	Set the monitor to generate a periodic event notification	
	2. Charging Station responds with **SetVariableMonitoringResponse**	**1.** OCTT sends **SetVariableMonitoringRequest** with: - **setMonitoringData[0].value** = *<Configured Clock Aligned MeterValues Interval>* - **setMonitoringData[0].type** = *Periodic* - **setMonitoringData[0].component.name** = *"EVSE"* - **setMonitoringData[0].component.evse.id** = *<Configured evseId>* - **setMonitoringData[0].variable.name** = *"AvailabilityState"*
	3. Charging Station generates **NotifyEventRequest** for EVSE #1::_AvailabilityState_ every *<Configured Clock Aligned MeterValues Interval>* seconds.	
Tool validations	* Step 2: Message: **SetVariableMonitoringResponse** with: **setMonitoringResult[0].status** = *Accepted* **setMonitoringResult[0].component.name** = *"EVSE"* **setMonitoringResult[0].component.evse.id** = *<Configured evseId>* **setMonitoringResult[0].variable.name** = *"AvailabilityState"* **setMonitoringResult[0].attributeStatusInfo** is absent or **attributeStatusInfo.reasonCode** = *"NoError"*	
	* Step 3: Message: a **NotifyEventRequest** message every *<Configured Clock Aligned MeterValues Interval>* seconds with: with an **eventData** element with: - **trigger** = *Periodic* - **component.name** = *"EVSE"* - **component.evse.id** = *1* - **variable.name** = *"AvailabilityState"*	
	Post scenario validations: N/A	

Table 358. Test Case Id: TC_N_25_CS

Test case name	**Retrieve Log Information - Diagnostics Log - Success**	
Test case Id	TC_N_25_CS	
Use case Id(s)	N01	
Requirement(s)	N01.FR.01, N01.FR.02, N01.FR.04, N01.FR.07, N01.FR.08, N01.FR.09, N01.FR.13	
System under test	Charging Station	
Description	This test case covers the functionality of getting log information from a Charging Station. The CSMS can request a Charging Station to upload a file with log information to a given location (URL). The format of this log file is not prescribed. The Charging Station successfully uploads a log file and gives information about the status of the upload by sending status notifications to the CSMS.	
Purpose	To verify if the Charging station is able to successfully upload a log as described at the OCPP specification.	
Prerequisite(s)	- Charging Station has log information available. - A diagnostics logging server has been setup supporting one of the file transfer protocols supported by the Charging Station (This is configured at the configuration variable **FileTransferProtocols**).	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **GetLogResponse**	**1.** The OCTT sends a **GetLogRequest** with **logType** *DiagnosticsLog*
	Note(s): *- Charging Station is uploading log file*	
	3. The Charging Station sends a **LogStatusNotificationRequest**	**4.** The OCTT responds with a **LogStatusNotificationResponse**
	Note(s): *- Log file is uploaded*	
	5. The Charging Station sends a **LogStatusNotificationRequest**	**6.** The OCTT responds with a **LogStatusNotificationResponse**
Tool validations	* Step 2: Message **GetLogResponse** - **status** *Accepted* - **filename** *not omitted AND not empty* * Step 3: Message **LogStatusNotificationRequest** - **status** *Uploading* - **requestId** *Same Id as the GetLogRequest* * Step 5: Message **LogStatusNotificationRequest** - **status** *Uploaded* - **requestId** *Same Id as the GetLogRequest*	
	Post scenario validations: - N/a	

Table 359. Test Case Id: TC_N_26_CS

Test case name	**Retrieve Log Information - Diagnostics Log - Upload failed**	
Test case Id	TC_N_26_CS	
Use case Id(s)	N01	
Requirement(s)	N01.FR.01, N01.FR.02, N01.FR.04, N01.FR.07, N01.FR.08, N01.FR.10, N01.FR.13	
System under test	Charging Station	
Description	This test case covers the functionality of getting log information from a Charging Station. The CSMS can request a Charging Station to upload a file with log information to a given location (URL). The format of this log file is not prescribed. The Charging Station unsuccessfully uploads a log file and gives information about the status of the upload by sending status notifications to the CSMS.	
Purpose	To verify if the Charging Station is able to correctly communicate with the CSMS after failing to upload a log as described at the OCPP specification.	
Prerequisite(s)	- Charging Station has log information available.	
Before (Preparations)	**Configuration State:** The retry interval should be configured longer than the time it takes to attempt an upload.	
	Memory State: Charging Station has log information available.	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **GetLogResponse**	**1.** The OCTT sends a **GetLogRequest** with - **logType** *DiagnosticsLog* - **retries** *3* - **retryInterval** <Configured retryInterval>
	3. The Charging Station sends a **LogStatusNotificationRequest**	**4.** The OCTT responds with a **LogStatusNotificationResponse**
	5. The Charging Station sends a **LogStatusNotificationRequest**	**6.** The OCTT responds with a **LogStatusNotificationResponse**
	Note(s): *- Steps 3 & 4 are optional after the first attempt.* *- The Charging Station will perform step (3,) 5, four times with <Configured retryInterval> seconds in between.*	

Test case name	**Retrieve Log Information - Diagnostics Log - Upload failed**
Tool validations	* Step 2: Message **GetLogResponse** - **status** *Accepted* * Step 3: Message **LogStatusNotificationRequest** - **status** *Uploading* - **requestId** *Same Id as the GetLogRequest* * Step 5: Message **LogStatusNotificationRequest** - **status** *UploadFailure* - **requestId** *Same Id as the GetLogRequest* OR Message **LogStatusNotificationRequest** - **status** *BadMessage* - **requestId** *Same Id as the GetLogRequest* OR Message **LogStatusNotificationRequest** - **status** *PermissionDenied* - **requestId** *Same Id as the GetLogRequest* OR Message **LogStatusNotificationRequest** - **status** *NotSupportedOperation* - **requestId** *Same Id as the GetLogRequest* * The time between the first **LogStatusNotificationRequest** *Uploading* and the last **LogStatusNotificationRequest** *UploadFailure/BadMessage/PermissionDenied/NotSupportedOperation* equals *(3 * <Configured retryInterval>)*
	Post scenario validations: - N/a

Table 360. Test Case Id: TC_N_27_CS

Test case name	**Get Customer Information - Accepted + data**	
Test case Id	TC_N_27_CS	
Use case Id(s)	N09	
Requirement(s)	N09.FR.02, N09.FR.05	
System under test	Charging Station	
Description	The CSMS sends a message to the Charging Station to retrieve IdToken customer information, for example to be compliant with local privacy laws. The Charging Station notifies the CSMS by sending one or more reports.	
Purpose	To verify if the Charging Station accepts the request and correctly sends the information as described at the OCPP specification.	
Prerequisite(s)	- The Charging Station needs to support Local Authorization and either the Local Authorization List or Authorization Cache. - The Charging Station supports authorization methods other than NoAuthorization	
Before (Preparations)	**Configuration State:** **LocalAuthListCtrlr.Enabled** is set to *true* **AuthCtrlr.LocalPreAuthorize** is set to *true* **AuthCacheCtrlr.Enabled** is set to *true*	
	Memory State: *IdTokenCached* for *<Configured valid IdToken fields>* (If implemented) *IdTokenLocalAuthList* for *<Configured valid IdToken fields>* (If implemented)	
	Charging State: **State** is *Authorized* (local) **State** is *ParkingBayUnoccupied*	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **CustomerInformationResponse**	**1.** The OCTT sends a **CustomerInformationRequest** with - **report** *true* - **idToken** <Configured valid idToken fields>
	3. The Charging Station sends a **NotifyCustomerInformationRequest**	**4.** The OCTT responds with a **NotifyCustomerInformationResponse** .
	Note(s): *- If **tbc** is True at Step 3 then step 3 and 4 will be repeated*	
Tool validations	* Step 2: Message **CustomerInformationResponse** - **status** *Accepted* * Step 3: Message **NotifyCustomerInformationRequest** - **data** *Not empty*	
	Post scenario validations: - All report parts have been received	

Table 361. Test Case Id: TC_N_28_CS

Test case name	**Get Customer Information - Accepted + no data**	
Test case Id	TC_N_28_CS	
Use case Id(s)	N09	
Requirement(s)	N09.FR.02, N09.FR.06	
System under test	Charging Station	
Description	The CSMS sends a message to the Charging Station to retrieve IdToken customer information, for example to be compliant with local privacy laws. The Charging Station notifies the CSMS by sending one or more reports.	
Purpose	To verify if the Charging Station accepts the request and correctly respond when it couldn't find the right information as described at the OCPP specification.	
Prerequisite(s)	Charging Station has no customer information available of <Configured valid idToken fields>	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: The CSMS requests the CS to clear the customerInformation for **idToken** <Configured valid idToken fields>	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **CustomerInformationResponse**	**1.** The OCTT sends a **CustomerInformationRequest** with - **report** *true* - **idToken** <Configured valid idToken fields>
	3. The Charging Station sends a **NotifyCustomerInformationRequest**	**4.** The OCTT responds with a **NotifyCustomerInformationResponse** .
Tool validations	* Step 2: Message **CustomerInformationResponse** - **status** *Accepted* * Step 3: Message **NotifyCustomerInformationRequest** - **tbc** *Not true*	
	Post scenario validations: - A message is sent indicating that no data is found	

Table 362. Test Case Id: TC_N_29_CS

Test case name	**Get Customer Information - Not Accepted**	
Test case Id	TC_N_29_CS	
Use case Id(s)	N09	
Requirement(s)	N09.FR.03	
System under test	Charging Station	
Description	The CSMS sends a message to the Charging Station to retrieve IdToken customer information, for example to be compliant with local privacy laws. The Charging Station notifies the CSMS by sending one or more reports.	
Purpose	To verify if the Charging Station correctly responds when it cannot process the request as described at the OCPP specification.	
Prerequisite(s)	Charging station is in a state where it cannot process customer information requests	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **CustomerInformationResponse**	**1.** The OCTT sends a **CustomerInformationRequest**
Tool validations	* Step 2: Message **CustomerInformationResponse** - **status** *Invalid*	
	Post scenario validations: - N/a	

Table 363. Test Case Id: TC_N_30_CS

Test case name	**Clear Customer Information - Clear and report + data**	
Test case Id	TC_N_30_CS	
Use case Id(s)	N10	
Requirement(s)	N10.FR.01, N10.FR.03	
System under test	Charging Station	
Description	The CSMS sends a message to the Charging Station to clear (and retrieve) IdToken customer information, for example to be compliant with local privacy laws. The Charging Station notifies the CSMS by sending one or more reports.	
Purpose	To verify if the Charging Station accepts the request and removes all customer related data (except from LocalList) and sent notifies as described at the OCPP specification.	
Prerequisite(s)	- The Charging Station needs to support Local Authorization and either the Local Authorization List or Authorization Cache. - The Charging Station supports authorization methods other than NoAuthorization	
Before (Preparations)	**Configuration State:** **LocalAuthListCtrlr.Enabled** is set to *true* **AuthCtrlr.LocalPreAuthorize** is set to *true* **AuthCacheCtrlr.Enabled** is set to *true*	
	Memory State: *IdTokenCached* for <*Configured valid IdToken fields*> (If implemented) *IdTokenLocalAuthList* for <*Configured valid IdToken fields*> (If implemented)	
	Charging State: **State** is *Authorized* (local) **State** is *ParkingBayUnoccupied*	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **CustomerInformationResponse**	**1.** The OCTT sends a **CustomerInformationRequest** with - **report** *true* AND - **clear** *true* AND - **idToken** <Configured valid idToken fields>
	3. The Charging Station sends a **NotifyCustomerInformationRequest**	**4.** The OCTT responds with a **NotifyCustomerInformationResponse** .
	Note(s): *- If* ***tbc*** *is True at Step 3 then step 3 and 4 will be repeated*	
	6. The Charging Station responds with a **CustomerInformationResponse**	**5.** The OCTT sends a **CustomerInformationRequest** with - **report** *true* AND - **idToken** <Configured valid idToken fields>
	7. The Charging Station sends a **NotifyCustomerInformationRequest**	**8.** The OCTT responds with a **NotifyCustomerInformationResponse** .
	Note(s): *- Step is optional and only expected when status is Accepted at Step 6*	

Test case name	**Clear Customer Information - Clear and report + data**
Tool validations	* Step 2: Message **CustomerInformationResponse** - **status** *Accepted* * Step 3: Message **NotifyCustomerInformationRequest** - **data** *Not empty* * Step 8:* Message **NotifyCustomerInformationRequest** - **tbc** *Not true*
	Post scenario validations: - All report parts have been received

Table 364. Test Case Id: TC_N_31_CS

Test case name	**Clear Customer Information - Clear and report + no data**	
Test case Id	TC_N_31_CS	
Use case Id(s)	N10	
Requirement(s)	N10.FR.01, N10.FR.04	
System under test	Charging Station	
Description	The CSMS sends a message to the Charging Station to clear (and retrieve) IdToken customer information, for example to be compliant with local privacy laws. The Charging Station notifies the CSMS by sending one or more reports.	
Purpose	To verify if the Charging Station accepts the request and correctly respond when it couldn't find the right information as described at the OCPP specification.	
Prerequisite(s)	Charging Station has no customer information available of <Configured valid idToken fields>	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **CustomerInformationResponse**	**1.** The OCTT sends a **CustomerInformationRequest** with - **report** *true* AND - **clear** *true* AND - **idToken** <Configured valid idToken fields>
	3. The Charging Station sends a **NotifyCustomerInformationRequest**	**4.** The OCTT responds with a **NotifyCustomerInformationResponse** .
Tool validations	* Step 2: Message **CustomerInformationResponse** - **status** *Accepted*	
	Post scenario validations: - A message is send indicating that no data is found	

Table 365. Test Case Id: TC_N_32_CS

Test case name	**Clear Customer Information - Clear and no report**	
Test case Id	TC_N_32_CS	
Use case Id(s)	N10	
Requirement(s)	N10.FR.01, N10.FR.06	
System under test	Charging Station	
Description	The CSMS sends a message to the Charging Station to clear (and retrieve) IdToken customer information, for example to be compliant with local privacy laws. The Charging Station notifies the CSMS by sending one or more reports.	
Purpose	To verify if the Charging Station accepts the request and removes all customer related data (except from LocalList) and sent one notify as described at the OCPP specification.	
Prerequisite(s)	Charging Station has no customer information available of <Configured valid idToken fields>	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **CustomerInformationResponse**	**1.** The OCTT sends a **CustomerInformationRequest** with - **report** *false* AND - **clear** *true* AND - **idToken** <Configured valid idToken fields>
	3. The Charging Station sends a **NotifyCustomerInformationRequest**	**4.** The OCTT responds with a **NotifyCustomerInformationResponse** .
Tool validations	* Step 2: Message **CustomerInformationResponse** - **status** *Accepted*	
	Post scenario validations: - A message is send indicating that the data is cleared	

Table 366. Test Case Id: TC_N_62_CS

Test case name	**Clear Customer Information - Clear and report - customerIdentifier**	
Test case Id	TC_N_62_CS	
Use case Id(s)	N10	
Requirement(s)	N10.FR.01, N10.FR.03	
System under test	Charging Station	
Description	The CSMS sends a message to the Charging Station to clear (and retrieve) raw customer information, for example to be compliant with local privacy laws. The Charging Station notifies the CSMS by sending one or more reports.	
Purpose	To verify if the Charging Station accepts the request and removes all customer related data (except from LocalList) and sent notifies as described at the OCPP specification.	
Prerequisite(s)	The Charging Station needs to support retrieving / deleting CustomerInformation - CustomerIdentifier.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: The tester needs manually store the *<Configured CustomerIdentifier>* at the Charging Station.	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **CustomerInformationResponse**	**1.** The OCTT sends a **CustomerInformationRequest** with - **report** *true* AND - **clear** *true* AND - **customerIdentifier** <Configured customerIdentifier>
	3. The Charging Station sends a **NotifyCustomerInformationRequest**	**4.** The OCTT responds with a **NotifyCustomerInformationResponse**
	Note(s): *- If **tbc** is True at Step 3 then step 3 and 4 will be repeated*	
	6. The Charging Station responds with a **CustomerInformationResponse**	**5.** The OCTT sends a **CustomerInformationRequest** with - **report** *true* AND - **clear** *false* AND - **customerIdentifier** <Configured customerIdentifier>
	7. The Charging Station sends a **NotifyCustomerInformationRequest**	**8.** The OCTT responds with a **NotifyCustomerInformationResponse**
	Note(s): *- If **tbc** is True at Step 7 then step 7 and 8 will be repeated*	
Tool validations	* Step 2: Message **CustomerInformationResponse** - **status** *Accepted* * Step 3: Message **NotifyCustomerInformationRequest** - **data** *Not empty* * Step 6: Message **CustomerInformationResponse** - **status** *Accepted* * Step 7: Message **NotifyCustomerInformationRequest** - **data** *empty*	
	Post scenario validations: - All report parts have been received	

Table 367. Test Case Id: TC_N_63_CS

Test case name	**Clear Customer Information - Clear and report - customerCertificate**	
Test case Id	TC_N_63_CS	
Use case Id(s)	N10	
Requirement(s)	N10.FR.09	
System under test	Charging Station	
Description	The CSMS sends a message to the Charging Station to clear (and retrieve) customer certificate information, for example to be compliant with local privacy laws. The Charging Station notifies the CSMS by sending one or more reports. Note: The only customer certificate that could exist in a charging station is a PnC contract certificate, which should not remain in the charging station.	
Purpose	To verify if the Charging Station accepts the request and removes all customer related data and sent notifies as described at the OCPP specification.	
Prerequisite(s)	The Charging Station needs to support retrieving / deleting CustomerInformation - CustomerCertificate.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: Execute **Reusable State** *EVConnectedPreSession* Execute **Reusable State** *Authorized15118* Execute **Reusable State** *ParkingBayUnoccupied*	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **CustomerInformationResponse**	**1.** The OCTT sends a **CustomerInformationRequest** with - **report** *true* AND - **clear** *true* AND - **customerCertificate** *customer information used in the transaction*
	3. The Charging Station sends a **NotifyCustomerInformationRequest**	**4.** The OCTT responds with a **NotifyCustomerInformationResponse**
	Note(s): *- If **tbc** is True at Step 3 then step 3 and 4 will be repeated*	
	6. The Charging Station responds with a **CustomerInformationResponse**	**5.** The OCTT sends a **CustomerInformationRequest** with - **report** *true* AND - **clear** *false* AND - **customerCertificate** *customer information used in the transaction*
	7. The Charging Station sends a **NotifyCustomerInformationRequest**	**8.** The OCTT responds with a **NotifyCustomerInformationResponse**
	Note(s): *- If **tbc** is True at Step 7 then step 7 and 8 will be repeated*	

Test case name	**Clear Customer Information - Clear and report - customerCertificate**
Tool validations	* Step 2: Message **CustomerInformationResponse** - **status** *Accepted* * Step 3: Message **NotifyCustomerInformationRequest** - **data** *empty* or *Not empty* if a customer certificate exists * Step 6: Message **CustomerInformationResponse** - **status** *Accepted* * Step 7: Message **NotifyCustomerInformationRequest** - **data** *empty*
	Post scenario validations: - All report parts have been received

Table 368. Test Case Id: TC_N_33_CS

Test case name	**Clear Customer Information - Invalid**	
Test case Id	TC_N_33_CS	
Use case Id(s)	N10	
Requirement(s)	N10.FR.01, N10.FR.05	
System under test	Charging Station	
Description	The CSMS sends a message to the Charging Station to clear (and retrieve) IdToken customer information, for example to be compliant with local privacy laws. The Charging Station notifies the CSMS by sending one or more reports.	
Purpose	To verify if the Charging Station rejects the request when it cannot process as described at the OCPP specification.	
Prerequisite(s)	Charging station is in a state where it cannot process customer information requests	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **CustomerInformationResponse**	**1.** The OCTT sends a **CustomerInformationRequest**
Tool validations	* Step 2: Message **CustomerInformationResponse** - **status** *Invalid*	
	Post scenario validations: - N/a	

Table 369. Test Case Id: TC_N_34_CS

Test case name	**Retrieve Log Information - Rejected**	
Test case Id	TC_N_34_CS	
Use case Id(s)	N01	
Requirement(s)	N01.FR.05	
System under test	Charging Station	
Description	This test case covers the functionality of getting log information from a Charging Station. The CSMS can request a Charging Station to upload a file with log information to a given location (URL). The format of this log file is not prescribed. The Charging Station successfully uploads a log file and gives information about the status of the upload by sending status notifications to the CSMS.	
Purpose	To verify if the Charging station is able to reject the request when no information is available as described at the OCPP specification.	
Prerequisite(s)	This testcase can only be executed if it is possible to have no log information available at the Charging Station.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **GetLogResponse**	**1.** The OCTT sends a **GetLogRequest** with **logType** *<Configured logType>*
Tool validations	* Step 2: Message **GetLogResponse** - **status** *Rejected*	
	Post scenario validations: - N/a	

Table 370. Test Case Id: TC_N_35_CS

Test case name	**Retrieve Log Information - Security Log - Success**	
Test case Id	TC_N_35_CS	
Use case Id(s)	N01	
Requirement(s)	N01.FR.01, N01.FR.02, N01.FR.03, N01.FR.07, N01.FR.08, N01.FR.09, N01.FR.13	
System under test	Charging Station	
Description	This test case covers the functionality of getting log information from a Charging Station. The CSMS can request a Charging Station to upload a file with log information to a given location (URL). The format of this log file is not prescribed. The Charging Station successfully uploads a log file and gives information about the status of the upload by sending status notifications to the CSMS.	
Purpose	To verify if the Charging station is able to successfully upload a log as described at the OCPP specification.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: Charging Station has log information available.	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **GetLogResponse**	**1.** The OCTT sends a **GetLogRequest** with **logType** *SecurityLog*
	Note(s): - *Charging Station is uploading log file*	
	3. The Charging Station sends a **LogStatusNotificationRequest**	**4.** The OCTT responds with a **LogStatusNotificationResponse** .
	Note(s): - *Log file is uploaded*	
	5. The Charging Station sends a **LogStatusNotificationRequest**	**6.** The OCTT responds with a **LogStatusNotificationResponse** .
Tool validations	* Step 2: Message **GetLogResponse** - **status** *Accepted* * Step 3: Message **LogStatusNotificationRequest** - **status** *Uploading* - **requestId** *Same Id as the GetLogRequest* * Step 5: Message **LogStatusNotificationRequest** - **status** *Uploaded* - **requestId** *Same Id as the GetLogRequest*	
	Post scenario validations: - N/a	

Table 371. Test Case Id: TC_N_36_CS

Test case name	**Retrieve Log Information - Second Request**	
Test case Id	TC_N_36_CS	
Use case Id(s)	N01	
Requirement(s)	N01.FR.01, N01.FR.02, N01.FR.03, N01.FR.07, N01.FR.08, N01.FR.09, N01.FR.12, N01.FR.13	
System under test	Charging Station	
Description	This test case covers the functionality of getting log information from a Charging Station. The CSMS can request a Charging Station to upload a file with log information to a given location (URL). The format of this log file is not prescribed. The Charging Station successfully uploads a log file and gives information about the status of the upload by sending status notifications to the CSMS.	
Purpose	To verify if the Charging station is able to successfully start/cancel a upload on a second request as described at the OCPP specification.	
Prerequisite(s)	The Charging Station supports cancelling an ongoing log file upload.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: Charging Station has log information available of *<Configured logType>*.	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **GetLogResponse**	**1.** The OCTT sends a **GetLogRequest** with **logType** *<Configured logType>*
	Note(s): *- Charging Station is uploading log file*	
	3. The Charging Station sends a **LogStatusNotificationRequest**	**4.** The OCTT responds with a **LogStatusNotificationResponse** .
	Note(s): *- Charging Station cancels uploading the first log file*	
	6. The Charging Station responds with a **GetLogResponse**	**5.** The OCTT sends a **GetLogRequest** with **logType** *<Configured logType>*
	7. The Charging Station sends a **LogStatusNotificationRequest**	**8.** The OCTT responds with a **LogStatusNotificationResponse** .
	Note(s): *- Charging Station is uploading log file*	
	9. The Charging Station sends a **LogStatusNotificationRequest**	**10.** The OCTT responds with a **LogStatusNotificationResponse** .
	Note(s): *- Log file is uploaded*	
	11. The Charging Station sends a **LogStatusNotificationRequest**	**12.** The OCTT responds with a **LogStatusNotificationResponse** .

Test case name	**Retrieve Log Information - Second Request**
Tool validations	* Step 2: Message **GetLogResponse** - **status** *Accepted* * Step 3: Message **LogStatusNotificationRequest** - **status** *Uploading* - **requestId** *Same Id as the GetLogRequest* * Step 6: Message **GetLogResponse** - **status** *AcceptedCanceled* * Step 7: Message **LogStatusNotificationRequest** - **status** *AcceptedCanceled* * Step 9: Message **LogStatusNotificationRequest** - **status** *Uploading* - **requestId** *Same Id as the GetLogRequest* * Step 11: Message **LogStatusNotificationRequest** - **status** *Uploaded* - **requestId** *Same Id as the GetLogRequest*
	Post scenario validations: - N/a

Table 372. Test Case Id: TC_N_37_CS

Test case name	**Set Variable Monitoring - Unknown Variable**	
Test case Id	TC_N_37_CS	
Use case Id(s)	N04	
Requirement(s)	N04.FR.04	
System under test	Charging Station	
Description	This test case describes how the CSMS requests the Charging Station to set monitoring triggers on Variables. Multiple triggers can be set for upper or lower thresholds, delta changes or periodic reporting.	
Purpose	To verify if the Charging station is able to correctly respond to the request when an unknown variable is sent as described at the OCPP specification.	
Prerequisite(s)	Charging Station supports Monitoring	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SetVariableMonitoringResponse**	**1.** The OCTT sends a **SetVariableMonitoringRequest** with **setMonitoringData.type** *Delta* **setMonitoringData.variable.name** *unknownVariable* **setMonitoringData.component.name** *EVSE*
Tool validations	* Step 2: Message **SetVariableMonitoringResponse** - **setMonitoringResult[0].status** *UnknownVariable* - **setMonitoringResult[0].type** *Delta* - **setMonitoringResult[0].severity** *<Configured severity>* - **setMonitoringResult[0].component.name** *EVSE* - **setMonitoringResult[0].variable.name** *unkownVariable*	
	Post scenario validations: - N/a	

Table 373. Test Case Id: TC_N_38_CS

Test case name	**Set Variable Monitoring - Not supported MonitorType**	
Test case Id	TC_N_38_CS	
Use case Id(s)	N04	
Requirement(s)	N04.FR.05	
System under test	Charging Station	
Description	This test case describes how the CSMS requests the Charging Station to set monitoring triggers on Variables. Multiple triggers can be set for upper or lower thresholds, delta changes or periodic reporting.	
Purpose	To verify if the Charging station is able to correctly respond to the request when a not supported monitortype is sent as described at the OCPP specification.	
Prerequisite(s)	- Charging Station supports Monitoring. - Charging station does not support one or more variableMonitoringTypes.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SetVariableMonitoringResponse**	**1.** The OCTT sends a **SetVariableMonitoringRequest** with **setVariableData *setMonitoringData.type** *UpperThreshold* **setMonitoringData.variable.name** *AvailabilityState* **setMonitoringData.component.name** *EVSE*
Tool validations	* Step 2: Message **SetVariableMonitoringResponse** - **setMonitoringResult[0].status** *UnsupportedMonitorType* or Rejected - **setMonitoringResult[0].type** *UpperThreshold* - **setMonitoringResult[0].component.name** *EVSE* - **setMonitoringResult[0].variable.name** *AvailabilityState*	
	Post scenario validations: - N/a	

Table 374. Test Case Id: TC_N_39_CS

Test case name	**Set Variable Monitoring - Component/Variable combination does NOT correspond**	
Test case Id	TC_N_39_CS	
Use case Id(s)	N04	
Requirement(s)	N04.FR.16	
System under test	Charging Station	
Description	This test case describes how the CSMS requests the Charging Station to set monitoring triggers on Variables. Multiple triggers can be set for upper or lower thresholds, delta changes or periodic reporting.	
Purpose	To verify if the Charging station is able to correctly respond to the request when a Component/Variable combination which does NOT correspond is sent as described at the OCPP specification.	
Prerequisite(s)	Charging Station supports Monitoring	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: Variable monitor is already set with component.name = EVSE, variable.name = AvailabilityState, type = Delta	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SetVariableMonitoringResponse**	**1.** The OCTT sends a **SetVariableMonitoringRequest** with **setMonitoringData.type** *UpperThreshold* **setMonitoringData.variable.name** *Power* **setMonitoringData.component.name** *ChargingStation*
	4. The Charging Station responds with a **GetMonitoringReportResponse**	**3.** The OCTT sends a **GetMonitoringReportRequest** with - **requestId** *<Generated requestId>*
	5. The Charging Station sends a **NotifyMonitoringReportRequest**	**6.** The OCTT responds with a **NotifyMonitoringReportResponse** .
	Note(s): *- If **tbc** is True at Step 3 then step 3 and 4 will be repeated*	
Tool validations	* Step 2: Message **SetVariableMonitoringResponse** - **setMonitoringResult[0].status** *Rejected* - **setMonitoringResult[0].type** *UpperThreshold* - **setMonitoringResult[0].severity** *<Configured severity>* - **setMonitoringResult[0].component.name** *ChargingStation* - **setMonitoringResult[0].variable.name** *Power* * Step 4: Message **GetMonitoringReportResponse** - **status** *Accepted* * Step 5: Message **NotifyMonitoringReportRequest** - **monitor.component** *EVSE* - **monitor.variable** *AvailablitiyState*	
	Post scenario validations: - All report parts have been received	

Table 375. Test Case Id: TC_N_40_CS

Test case name	**Set Variable Monitoring - Replace Variable Monitor**	
Test case Id	TC_N_40_CS	
Use case Id(s)	N04	
Requirement(s)	N04.FR.12	
System under test	Charging Station	
Description	This test case describes how the CSMS requests the Charging Station to set monitoring triggers on Variables. Multiple triggers can be set for upper or lower thresholds, delta changes or periodic reporting.	
Purpose	To verify if the Charging station is able to correctly replace an existing variable monitor as described at the OCPP specification.	
Prerequisite(s)	Charging Station supports Monitoring	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: Variable monitor is set for *Delta* with severity *5*	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SetVariableMonitoringResponse**	**1.** The OCTT sends a **SetVariableMonitoringRequest** with **setMonitoringData.id** *<Generated variableMonitoringId>* AND **setMonitoringData.type** *Delta* **setMonitoringData.severity** *4*
	4. The Charging Station responds with a **GetMonitoringReportResponse**	**3.** The OCTT sends a **GetMonitoringReportRequest** with - **requestId** *<Generated requestId>* - **componentVariable.component.name** *EVSE* - **componentVariable.component.evse.id** *evseId* - **componentVariable.variable.name** *AvailabilityState* - **monitoringCriteria** *DeltaMonitoring_*
	5. The Charging Station sends a **NotifyMonitoringReportRequest**	**6.** The OCTT responds with a **NotifyMonitoringReportResponse** .
Tool validations	* Step 2: Message **SetVariableMonitoringResponse** - **setMonitoringResult[0].status** *Accepted* - **setMonitoringResult[0].type** *Delta* - **setMonitoringResult[0].component.name** *EVSE* - **setMonitoringResult[0].variable.name** *AvailabilityState* * Step 4: Message **GetMonitoringReportResponse** - **status** *Accepted* * Step 5: Message **NotifyMonitoringReportRequest** - **monitor.component.name** *EVSE* - **monitor.variable.name** *AvailabilityState* - **monitor.variableMonitoring.severity** *4*	
	Post scenario validations: - All report parts have been received	

Table 376. Test Case Id: TC_N_41_CS

Test case name	**Set Variable Monitoring - Return to FactoryDefault**	
Test case Id	TC_N_41_CS	
Use case Id(s)	N03	
Requirement(s)	N03.FR.04, N04.FR.15	
System under test	Charging Station	
Description	This test case describes how the CSMS requests the Charging Station to overrule a preconfigured monitor by a custom monitor. When monitoringBase is set to FactoryDefault the preconfigured monitor must return.	
Purpose	To verify if the Charging station is able to correctly restore monitors to FactoryDefault.	
Prerequisite(s)	Charging Station supports Monitoring	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: A preconfigured monitor exists with *id <Preconfigured monitor id>* for *component EVSE* and *variable AvailabilityState* and *type = Delta* and *severity = <Preconfigured severity>*	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SetVariableMonitoringResponse**	**1.** The OCTT sends a **SetVariableMonitoringRequest** with **setMonitoringData.id** *<Preconfigured monitor id>* AND **setMonitoringData.type** *Delta* **setMonitoringData.severity** *<Preconfigured severity> + 1*
	4. The Charging Station responds with a **GetMonitoringReportResponse**	**3.** The OCTT sends a **GetMonitoringReportRequest** with - **requestId** *<Generated requestId>* - **id** *<Preconfigured monitor id>* - ***componentVariable.component.name*** *_EVSE* - **componentVariable.component.evse.id** *evseId* - **componentVariable.variable.name** *AvailabilityState* - **monitoringCriteria** *DeltaMonitoring*
	5. The Charging Station sends a **NotifyMonitoringReportRequest**	**6.** The OCTT responds with a **NotifyMonitoringReportResponse** .
	8. The Charging Station responds with a **SetMonitoringBaseResponse** with - **status** *Accepted*	**7.** The OCTT sends a **SetMonitoringBaseRequest** with - **monitoringBase** *FactoryDefault*
	10. The Charging Station responds with a **GetMonitoringReportResponse**	**9.** The OCTT sends a **GetMonitoringReportRequest** with - **requestId** *<Generated requestId>* - **id** *<Preconfigured monitor id>* - ***componentVariable.component.name*** *_EVSE* - **componentVariable.component.evse.id** *evseId* - **componentVariable.variable.name** *AvailabilityState* - **monitoringCriteria** *DeltaMonitoring*
	11. The Charging Station sends a **NotifyMonitoringReportRequest**	**12.** The OCTT responds with a **NotifyMonitoringReportResponse** .

Test case name	**Set Variable Monitoring - Return to FactoryDefault**
Tool validations	* Step 2: Message **SetVariableMonitoringResponse** - **setMonitoringResult[0].status** *Accepted* - **setMonitoringResult[0].type** *Delta* - **setMonitoringResult[0].component.name** *EVSE* - **setMonitoringResult[0].variable.name** *AvailabilityState* * Step 4: Message **GetMonitoringReportResponse** - **status** *Accepted* * Step 5: Message **NotifyMonitoringReportRequest** - **monitor.component.name** *EVSE* - **monitor.variable.name** *AvailabilityState* - **monitor.variableMonitoring.id** *<Preconfigured id>* - **monitor.variableMonitoring.severity** *<Preconfigured severity> + 1* * Step 11: Message **NotifyMonitoringReportRequest** - **monitor.component.name** *EVSE* - **monitor.variable.name** *AvailabilityState* - **monitor.variableMonitoring.id** *<Preconfigured id>* - **monitor.variableMonitoring.severity** *<Preconfigured severity>*
	Post scenario validations: - All report parts have been received

Table 377. Test Case Id: TC_N_43_CS

Test case name	**Set Variable Monitoring - First SetMonitoringData and third SetMonitoringData are valid, but the second contains an out of range value**	
Test case Id	TC_N_43_CS	
Use case Id(s)	N04	
Requirement(s)	N/a	
System under test	Charging Station	
Description	This test case describes how the CSMS requests the Charging Station to set monitoring triggers on Variables. Multiple triggers can be set for upper or lower thresholds, delta changes or periodic reporting.	
Purpose	To verify if the Charging station is able to correctly respond when one of requested variable monitor data is out of range replace as described at the OCPP specification.	
Prerequisite(s)	Charging Station supports Monitoring	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SetVariableMonitoringResponse**	**1.** The OCTT sends a **SetVariableMonitoringRequest** with - **setMonitoringData.component.name** = *<Configured threshold monitor component variable>* - **setMonitoringData.variable.name** = *<Configured threshold monitor component variable>* - **setMonitoringData[0].value** = *<Configured threshold monitor value>* - **setMonitoringData[0].type** = *UpperThreshold* - **setMonitoringData[1].value** = *-1.0* - **setMonitoringData[1].type** = *Delta* - **setMonitoringData[2].value** = *<Configured threshold monitor2 value>* - **setMonitoringData[2].type** = *LowerThreshold*

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

	Post scenario validations: - N/a	

Table 378. Test Case Id: TC_N_44_CS

Test case name	**Clear Monitoring - Rejected**	
Test case Id	TC_N_44_CS	
Use case Id(s)	N06	
Requirement(s)	N06.FR.03	
System under test	Charging Station	
Description	A monitoring setting can be cleared (removed) by sending a ClearVariableMonitoringRequest with the id of the monitoring setting.	
Purpose	To verify if the Charging station is able to correctly respond on a request to clear a monitor that cannot be cleared as described at the OCPP specification.	
Prerequisite(s)	Charging Station supports Monitoring, Charging Station has hard-coded monitor(s)	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: OCTT Sends a GetMonitoringReportRequest, the CS then reports all existsing monitors if it has any. These monitors should be hard-coded and the first Id is used fot the TC.	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **ClearVariableMonitoringResponse**	**1.** The OCTT sends a **ClearVariableMonitoringRequest** with **id** *monitor id from the Preparations*
Tool validations	* Step 2: Message **ClearVariableMonitoringResponse** - **clearMonitoringResult[0].status** *Rejected*	
	Post scenario validations: - N/a	

Table 379. Test Case Id: TC_N_45_CS

Test case name	**Alert Event - Delta value exceeded**	
Test case Id	TC_N_45_CS	
Use case Id(s)	N07	
Requirement(s)	N07.FR.06, N07.FR.07, N07.FR.18, N07.FR.19	
System under test	Charging Station	
Description	NotifyEventRequest reports every Component/Variable for which a VariableMonitoring setting was triggered. Only the VariableMonitoring settings that are responsible for triggering an event are included.	
Purpose	To verify if the Charging station is correctly communicating when a delta value has exceeded as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: Variable monitor is configured with: - **setMonitoringData.component.name** = *<Configured threshold monitor component variable>* - **setMonitoringData.component.evse.id** = *<Configured EVSEId>* - **setMonitoringData.value** = *<Configured threshold monitor value>* - **setMonitoringData.type** = *Delta* - **setMonitoringData.variable.name** = *<Configured delta monitor component variable>* Notes: *If componentVariable is set to "Power" or "Current", the value is set to 100.0*	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *If componentVariable is set to "Power" or "Current" EnergyTransferStarted will trigger the monitor. If another componentvariable is chosen a manual action is needed to trigger the monitor.*	
	1. Execute **Reusable State** *EnergyTransferStarted* or manually trigger the monitor.	
	2. The Charging Station sends a **NotifyEventRequest**	**3.** The OCTT responds with a **NotifyEventResponse** .
	Note(s): - *If **tbc** is True at Step 2 then step 1 and 3 will be repeated*	
Tool validations	* Step 2: Message **NotifyEventRequest** - **eventData[0].trigger** *Delta* - **eventData[0].component.name** *<Configured threshold monitor component variable>* - **eventData[0].variable.name** *<Configured threshold monitor component variable>* - **eventData[0].variableMonitoringId** *<Configured variableMonitoringId>*	
	Post scenario validations: - N/a	

Table 380. Test Case Id: TC_N_47_CS

Test case name	**Get Monitoring report - Report all**	
Test case Id	TC_N_47_CS	
Use case Id(s)	N02	
Requirement(s)	N02.FR.01, N02.FR.11	
System under test	Charging Station	
Description	This test case describes how the CSMS requests the Charging Station to send a report about configured monitoring settings per component and variable. Optionally, this list can be filtered on monitoringCriteria and componentVariables.	
Purpose	To verify if the Charging station is able to correctly report all monitoring data as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: The following monitors must be present as 'hard-wired' or 'preconfigured' or must have been configured by CSMS: - Component "ChargingStation", variable "AvailabilityState", monitor type *Delta*" - Component "EVSE", Configured evse, variable "AvailabilityState", monitor type *Delta*	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **GetMonitoringReportResponse**	**1.** The OCTT sends a **GetMonitoringReportRequest** with **monitoringCriteria** omitted AND **componentVariable** omitted.
	3. The Charging Station sends a **NotifyMonitoringReportRequest**	**4.** The OCTT responds with a **NotifyMonitoringReportResponse** .
	Note(s): *- If **tbc** is True at Step 3 then step 3 and 4 will be repeated*	
Tool validations	* Step 3: Message: **NotifyMonitoringReportRequest** - **requestId** = *<Generated requestId>* While **tbc** = *true* , Message: **NotifyMonitoringReportRequest** - **monitor.variable** = *"AvailabilityState"* - **monitor.variableMonitoring.type** = *Delta* - **monitor.component_.name** = *ChargingStation* or *EVSE*	
	Post scenario validations: - All reports have been received	

Table 381. Test Case Id: TC_N_48_CS

Test case name	**Alert Event - Variable monitoring on write only**	
Test case Id	TC_N_48_CS	
Use case Id(s)	N07	
Requirement(s)	N07.FR.10	
System under test	Charging Station	
Description	NotifyEventRequest reports every Component/Variable for which a VariableMonitoring setting was triggered. Only the VariableMonitoring settings that are responsible for triggering an event are included.	
Purpose	To verify if the Charging station is able to correctly omit the actualField when a variablemonitor has been set to write only as described at the OCPP specification.	
Prerequisite(s)	The Charging Station should be able to set a monitor on SecurityCtrlr.BasicAuthPassword and should be able to use security profile 1 or 2	
Before (Preparations)	**Configuration State:** Security profile 1 or 2 is configured	
	Memory State: A Delta variableMonitoring setting has been set on a SecurityCtrlr.BasicAuthPassword	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SetVariablesResponse** .	**1.** The OCTT sends a **SetVariablesRequest** with **component.name** = *SecurityCtrlr* **variable.name** = *BasicAuthPassword* **attributeValue** = *<Generated password with same length as the configured basicAuthPassword>*
	3. Execute **Reusable State** *Booted*. Notes: *This step only needs to be executed when **SetVariablesResponse** status is RebootRequired.*	
	4. The Charging station sends a **NotifyEventRequest**	**5.** The OCTT responds with a **NotifyEventResponse** .
Tool validations	* Step 2: Message **SetVariablesResponse** - **status** must be *Accepted* or *RebootRequired* * Step 4: Message **NotifyEventRequest** - **eventData[0].actualValue** must be an empty string	
	Post scenario validations: - N/a	

Table 382. Test Case Id: TC_N_61_CS

Test case name	**Alert Event - Variable monitoring on numeric**	
Test case Id	TC_N_61_CS	
Use case Id(s)	N07	
Requirement(s)	N07.FR.10	
System under test	Charging Station	
Description	NotifyEventRequest reports every Component/Variable for which a VariableMonitoring setting was triggered. Only the VariableMonitoring settings that are responsible for triggering an event are included.	
Purpose	To verify if the Charging station is able to correctly respond when a nomeric Delta monitor is matched and exceeded, as described at the OCPP specification.	
Prerequisite(s)	The Charging Station should be able to set a monitor on OCPPCommCtrlr.OfflineThreshold	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: A Delta variableMonitoring setting has been set on a OCPPCommCtrlr.OfflineThreshold	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SetVariablesResponse** .	**1.** The OCTT sends a **SetVariablesRequest** with **component.name** = *OCPPCommCtrlr* **variable.name** = *OfflineThreshold* **attributeValue** = *Current Threshold + 1*
	3. Execute **Reusable State** *Booted*. Notes: *This step only needs to be executed when **SetVariablesResponse** status is RebootRequired.*	
	Notes: *The CS should not send a NotifyEvent as the delta monitor was not exceeded.*	
	5. The Charging Station responds with a **SetVariablesResponse** .	**4.** The OCTT sends a **SetVariablesRequest** with **component.name** = *OCPPCommCtrlr* **variable.name** = *OfflineThreshold* **attributeValue** = *Current Threshold + 2*
	6. Execute **Reusable State** *Booted*. Notes: *This step only needs to be executed when **SetVariablesResponse** status is RebootRequired.*	
	7. The Charging station sends a **NotifyEventRequest**	**8.** The OCTT responds with a **NotifyEventResponse** .
Tool validations	* Step 2: Message **SetVariablesResponse** - **status** must be *Accepted* or *RebootRequired*+ * Step 5: Message **SetVariablesResponse** - **status** must be *Accepted* or *RebootRequired*+ * Step 7: Message **NotifyEventRequest** - **eventData[0].actualValue** must be *Current Threshold + 2*	
	Post scenario validations: - N/a	

Table 383. Test Case Id: TC_N_51_CS

Test case name	**Set Variable Monitoring - Replace Variable Monitor**	
Test case Id	TC_N_51_CS	
Use case Id(s)	N07	
Requirement(s)	N07.FR.11	
System under test	Charging Station	
Description	NotifyEventRequest reports every Component/Variable for which a VariableMonitoring setting was triggered. Only the VariableMonitoring settings that are responsible for triggering an event are included.	
Purpose	To verify if the Charging station is able to correctly check if the current value exceeds the new threshold as described at the OCPP specification.	
Prerequisite(s)	Charging Station supports Monitoring	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: Variable monitor is already set with: **setMonitoringData.component.name** *<Configured threshold monitor component variable>* AND **setMonitoringData.component.evse.id** *<Configured EVSEId>* AND **setMonitoringData.value** *<Configured threshold monitor value>* AND **setMonitoringData.type** *UpperThreshold* AND **setMonitoringData.variable.name** *<Configured threshold monitor component variable>* Notes: *If componentVariable is set to "Power" or "Current", the value is set to the configured maxLimit -1*	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	Notes: *If componentVariable is set to "Power" or "Current" EnergyTransferStarted will trigger the monitor. If another componentvariable is chosen a manual action is needed to trigger the monitor.*	
	1. Execute **Reusable State** *EnergyTransferStarted* or manually trigger the monitor.	
	3. The Charging Station responds with a **SetVariableMonitoringResponse**	**2.** The OCTT sends a **SetVariableMonitoringRequest** with **setMonitoringData.component.name** *<Configured threshold monitor component variable>* AND **setMonitoringData.component.evse.id** *<Configured EVSEId>* AND **setMonitoringData.id** *<Configured variableMonitoringId>* AND **setMonitoringData.value** *<Configured threshold monitor value2>* AND **setMonitoringData.type** *UpperThreshold* **setMonitoringData.variable.name** *<Configured threshold monitor component variable>* Notes: *If componentVariable is set to "Power" or "Current", the value is set to 0.0*
	4. The Charging station sends a **NotifyEventRequest**	**5.** The OCTT responds with a **NotifyEventResponse** .

Test case name	**Set Variable Monitoring - Replace Variable Monitor**
Tool validations	* Step 3: Message **SetVariableMonitoringResponse** - **setMonitoringResult[0].status** *Accepted* - **setMonitoringResult[0].type** *UpperThreshold* - **setMonitoringResult[0].severity** *<Configured severity>* - **setMonitoringResult[0].component.name** *<Configured threshold monitor component variable>* - **setMonitoringResult[0].variable.name** *<Configured threshold monitor component variable>* * Step 4: Message **NotifyEventRequest** - **eventData[0].trigger** *Alerting* - **eventData[0].actualValue** > *<Configured threshold monitor value>*
	Post scenario validations: - All report parts have been received

Table 384. Test Case Id: TC_N_52_CS

Test case name	**Set Variable Monitoring - Removing a VariableMonitor**	
Test case Id	TC_N_52_CS	
Use case Id(s)	N07	
Requirement(s)	N07.FR.12	
System under test	Charging Station	
Description	NotifyEventRequest reports every Component/Variable for which a VariableMonitoring setting was triggered. Only the VariableMonitoring settings that are responsible for triggering an event are included.	
Purpose	To verify if the Charging station is able to correctly communicate when a threshold has been exceeded and the applicable monitor is removed as described at the OCPP specification.	
Prerequisite(s)	Charging Station supports Monitoring	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: Variable monitor is already set with: **setMonitoringData.component.name** *<Configured threshold monitor component variable>* AND **setMonitoringData.component.evse.id** *<Configured EVSEId>* AND **setMonitoringData.value** *<Configured threshold monitor value>* AND **setMonitoringData.type** *UpperThreshold* AND **setMonitoringData.variable.name** *<Configured threshold monitor component variable>* Notes: *If componentVariable is set to "Power" or "Current", the value is set to 0.0*	
	Charging State: Execute **Reusable State** *EnergyTransferStarted* or manually trigger the monitor. Notes: *If componentVariable is set to "Power" or "Current" EnergyTransferStarted will trigger the monitor. If another componentvariable is chosen a manual action is needed to trigger the monitor.*	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **ClearVariableMonitoringResponse**	**1.** The OCTT sends a **ClearVariableMonitoringRequest** with **id** *<Configured variableMonitoringId>*
	4. The Charging Station responds with a **GetMonitoringReportResponse**	**3.** The OCTT sends a **GetMonitoringReportRequest** with **componentVariable.component** *<Configured threshold monitor component variable>* **componentVariable.variable** *<Configured threshold monitor component variable>* **monitoringCriteria** *ThresholdMonitoring*
	5. Execute **Reusable State** *StopAuthorized* or manually trigger the monitor. Notes: *If componentVariable is set to "Power" or "Current" EnergyTransferStarted will trigger the monitor. If another componentvariable is chosen a manual action is needed to trigger the monitor.*	
	6. The Charging Station should not send a request for the cleared monitor	
Tool validations	* Step 2: Message **ClearVariableMonitoringResponse** - **clearMonitoringResult[0].status** *Accepted* AND - **clearMonitoringResult[0].id** *<Configured variableMonitoringId>*	
	* Step 4: Message **GetMonitoringReportResponse** - **getMonitoringResult[0].status** *EmptyResultSet*	
	* Step 6: - No NotifyEventRequest with **variableMontioringId** *<Configured variableMonitoringId>* is send	
	Post scenario validations: - N/a	

Table 385. Test Case Id: TC_N_53_CS

Test case name	**Alert Event - Persistant over reboot**	
Test case Id	TC_N_53_CS	
Use case Id(s)	N07	
Requirement(s)	N07.FR.13	
System under test	Charging Station	
Description	NotifyEventRequest reports every Component/Variable for which a VariableMonitoring setting was triggered. Only the VariableMonitoring settings that are responsible for triggering an event are included.	
Purpose	To verify if the Charging station is able to save the variableMonitor data persistent across reboot as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: Variable monitor is already set with: **setMonitoringData.component.name** *<Configured threshold monitor component variable>* AND **setMonitoringData.component.evse.id** *<Configured EVSEId>* AND **setMonitoringData.value** *<Configured threshold monitor value>* AND **setMonitoringData.type** *UpperThreshold* AND **setMonitoringData.variable.name** *<Configured threshold monitor component variable>*	
	Charging State: Execute **Reusable State** *Booted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **GetMonitoringReportResponse**	**1.** The OCTT sends a **GetMonitoringReportRequest** with **monitoringCriteria** *ThresholdMonitoring*
	3. The Charging Station sends a **NotifyMonitoringReportRequest**	**4.** The OCTT responds with a **NotifyMonitoringReportResponse** .
	Note(s): - *If **tbc** is True at Step 3 then step 3 and 4 will be repeated*	
Tool validations	* Step 3: Message **NotifyMonitoringReportRequest** - **requestId** *<The Id of the request>* AND - **monitor.variableMonitoring.id** *<Received monitorId from set monitor>* - **monitor.variableMonitoring.type** *UpperThreshold*	
	Post scenario validations: - All reports have been received	

Table 386. Test Case Id: TC_N_56_CS

Test case name	**Alert Event - Delta value NOT numeric exceeded**	
Test case Id	TC_N_56_CS	
Use case Id(s)	N07	
Requirement(s)	N07.FR.06, N07.FR.07, N07.FR.18, N07.FR.19	
System under test	Charging Station	
Description	NotifyEventRequest reports every Component/Variable for which a VariableMonitoring setting was triggered. Only the VariableMonitoring settings that are responsible for triggering an event are included.	
Purpose	To verify if the Charging station is correctly communicating when a delta value has exceeded as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: Variable monitor is configured with: **component.evse.id** *<Configured EVSEId>* **component.name** *EVSE* **severity** *<Configured severity>* **type** *Delta* **value** *1.0* **variable.name** *AvailablityState*	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Make sure the configured delta value has been exceeded*	
	1. The Charging Station sends a **NotifyEventRequest**	**2.** The OCTT responds with a **NotifyEventResponse** .
	Note(s): *- If **tbc** is True at Step 1 then step 1 and 2 will be repeated*	
Tool validations	* Step 1: Message **NotifyEventRequest** - **eventData[0].trigger** *Delta* - **eventData[0].component.name** *EVSE* - **eventData[0].variable.name** *AvailabilityState* - **eventData[0].variableMonitoringId** *monitoringId of monitor set in Memory State*	
	Post scenario validations: - N/a	

2.16. O Display Message

Table 387. Test Case Id: TC_O_01_CS

Test case name	**Set Display Message - Success**	
Test case Id	TC_O_01_CS	
Use case Id(s)	O01	
Requirement(s)	O01_FR_12	
System under test	Charging Station	
Description	This test case describes how the CSMS can be requested to sent an SetDisplayMessageRequest to the charging station. Depending on the given parameters the message shall be displayed a certain way and at a certain moment on the Charging Station. These messages are displayed additionally on a Charging Station and are not part of the firmware.	
Purpose	To verify if the Charging Station is able to display additional messages according to the DisplayMessage mechanism as described in the OCPP specification.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SetDisplayMessageResponse**	**1.** The OCTT sends a **SetDisplayMessageRequest** with **message.id** *<Generated displayMessageId>* **message.priority** *<Configured priority>*
	Note(s): - *The display message is displayed as configured*	
	4. The Charging Station responds with a **GetDisplayMessagesResponse**	**3.** The OCTT sends a **GetDisplayMessagesRequest** with **id** *<Generated displayMessageId>* **requestId** *<Generated requestId>*
	5. The Charging Station sends a **NotifyDisplayMessagesRequest**	**6.** The OCTT responds with a **NotifyDisplayMessagesResponse** .
Tool validations	* Step 2: Message **SetDisplayMessageResponse** - **status** *Accepted* * Step 4: Message **GetDisplayMessagesResponse** - **status** *Accepted* * Step 5: Message **NotifyDisplayMessagesRequest** - **requestId** *<RequestId sent in step 3>* - **id** *<Generated id>* - **priority** *<Configured Priority>* - **message.format** *<Configured format>* - **message.content** *<Configured content>*	
	Post scenario validations: - N/a	

Table 388. Test Case Id: TC_O_02_CS

Test case name	**Get all Display Messages - Success**	
Test case Id	TC_O_02_CS	
Use case Id(s)	O03	
Requirement(s)	O03_FR_01, O03_FR_02, O03_FR_03, O03_FR_04, O03_FR_05	
System under test	Charging Station	
Description	This test case describes how a CSO can request all the installed DisplayMessages configured via OCPP in a Charging Station. The Charging Station can remove messages when they are out-dated, or transactions have ended. It can be very useful for a CSO to be able to view to current list of messages, so the CSO knows which messages are (still) configured.	
Purpose	To verify if the Charging Station is able to send the requested DisplayMessages according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: At least 1 display message is configured.	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **GetDisplayMessagesResponse**	**1.** The OCTT sends a **GetDisplayMessagesRequest requestId** *<Generated requestId>*
	3. The Charging Station sends a **NotifyDisplayMessagesRequest**	**4.** The OCTT responds with a **NotifyDisplayMessagesResponse** .
	Note(s): *- If **tbc** is True at Step 3 then step 3 and 4 will be repeated*	
Tool validations	* Step 2: Message **GetDisplayMessagesResponse** - **status** *Accepted* * Step 3: Message **NotifyDisplayMessagesRequest** - **requestId** *<Generated requestId>*	
	Post scenario validations: - All messages have been received	

Table 389. Test Case Id: TC_O_03_CS

Test case name	**Get all Display Messages - No DisplayMessages configured**	
Test case Id	TC_O_03_CS	
Use case Id(s)	O03	
Requirement(s)	O03_FR_06	
System under test	Charging Station	
Description	This test case describes how a CSO can request all the installed DisplayMessages configured via OCPP in a Charging Station. The Charging Station can remove messages when they are out-dated, or transactions have ended. It can be very useful for a CSO to be able to view to current list of messages, so the CSO knows which messages are (still) configured.	
Purpose	To verify if the Charging Station is responding according to the DisplayMessage mechanism as described in the OCPP specification when no Display Messages are configured.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **GetDisplayMessagesResponse**	**1.** The OCTT sends a **GetDisplayMessagesRequest**
Tool validations	* Step 2: Message **GetDisplayMessagesResponse** - **status** *Unknown*	
	Post scenario validations: - N/a	

Table 390. Test Case Id: TC_O_04_CS

Test case name	**Clear Display Message - Success**	
Test case Id	TC_O_04_CS	
Use case Id(s)	O05	
Requirement(s)	O05_FR_01	
System under test	Charging Station	
Description	This test case describes how a CSO can remove a specific message, configured via OCPP in a Charging Station.	
Purpose	To verify if the Charging Station is able to remove a specific message requested by the CSMS according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: A message with <Generated displayMessageId> is configured.	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **ClearDisplayMessageResponse**	**1.** The OCTT sends a **ClearDisplayMessageRequest** with **id** *<Generated displayMessageId>*
	4. The Charging Station responds with a **GetDisplayMessagesResponse**	**3.** The OCTT sends a **GetDisplayMessagesRequest** with **id** *<Generated displayMessageId>*
Tool validations	* Step 2: Message **ClearDisplayMessageResponse** - **status** *Accepted* * Step 4: Message: **GetDisplayMessagesResponse** - **status** must be *Unknown*	
	Post scenario validations: - N/a	

Table 391. Test Case Id: TC_O_05_CS

Test case name	**Clear Display Message - Unknown Key**	
Test case Id	TC_O_05_CS	
Use case Id(s)	O05	
Requirement(s)	O05_FR_02	
System under test	Charging Station	
Description	This test case describes how a CSO can remove a specific message, configured via OCPP in a Charging Station.	
Purpose	To verify if the Charging Station is able to respond according the mechanism as described in the OCPP specification when no message is configured with the specified id.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **ClearDisplayMessageResponse**	**1.** The OCTT sends a **ClearDisplayMessageRequest** with **id** *<Generated displayMessageId>*
Tool validations	* Step 2: Message **ClearDisplayMessageResponse** - **status** *Unknown*	
	Post scenario validations: - N/a	

Table 392. Test Case Id: TC_O_06_CS

Test case name	**Set Display Message - Specific transaction - Success**	
Test case Id	TC_O_06_CS	
Use case Id(s)	O02	
Requirement(s)	O02.FR.02, O02_FR_14	
System under test	Charging Station	
Description	This test case describes how a CSO can set a message to be displayed on a Charging Station for a specific transaction. Depending on the given parameters the message shall be displayed a certain way on the Charging Station.	
Purpose	To verify if the Charging Station is able to display the message correctly according the mechanism as described in the OCPP specification when a transaction is ongoing.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: **State** is *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SetDisplayMessageResponse**	**1.** The OCTT sends a **SetDisplayMessageRequest** with **message.id** *<Generated displayMessageId>* AND **message.transactionId** *<Configured transactionId>* AND **message.priority** *<Configured Priority*
	Note(s): - *The display message is displayed as configured*	
	3. Execute **Reusable State** *StopAuthorized*	
	4. Execute **Reusable State** *EVConnectedPostSession*	
	5. Execute **Reusable State** *EVDisconnected*	
	6. Execute **Reusable State** *ParkingBayUnoccupied*	
	Note(s): - *The display message is not displayed anymore*	
	8. The Charging Station responds with a **GetDisplayMessagesResponse**	**7.** The OCTT sends a **GetDisplayMessagesRequest** with **id** *<Generated displayMessageId>*
Tool validations	* Step 1: Message: **SetDisplayMessageResponse** - **status** must be *Accepted* * Step 8: Message: **GetDisplayMessagesResponse** - **status** must be *Unknown*	
	Post scenario validations: N/a	

Table 393. Test Case Id: TC_O_07_CS

Test case name	**Get a Specific Display Message - Id**	
Test case Id	TC_O_07_CS	
Use case Id(s)	O04	
Requirement(s)	O04_FR_01, O04_FR_03, O04_FR_04, O04_FR_05, O04_FR_06	
System under test	Charging Station	
Description	This test case describes how a CSO can request specific installed DisplayMessages configured via OCPP in a Charging Station. The Charging Station can remove messages when they are out-dated, or transactions have ended. It can be very useful for a CSO to be able to view to current list of messages, so the CSO knows which messages are (still) configured.	
Purpose	To verify if the Chargin Station is able to respond the specific id message requested by the CSMS according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: A display message with <Generated displayMessageId> is configured.	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **GetDisplayMessagesResponse**	**1.** The OCTT sends a **GetDisplayMessagesRequest** with **id** *<Generated displayMessageId>* **requestId** *<Generated requestId>*
	3. The Charging Station sends a **NotifyDisplayMessagesRequest**	**4.** The OCTT responds with a **NotifyDisplayMessagesResponse** .
	Note(s): *- If **tbc** is True at Step 3 then step 3 and 4 will be repeated*	
Tool validations	* Step 2: Message **GetDisplayMessagesResponse** - **status** *Accepted* * Step 3: Message **NotifyDisplayMessagesRequest** - **requestId** *<Generated requestId>*	
	Post scenario validations: - All messages have been received	

Table 394. Test Case Id: TC_O_08_CS

Test case name	**Get a Specific Display Message - Priority**	
Test case Id	TC_O_08_CS	
Use case Id(s)	O04	
Requirement(s)	O04_FR_01, O04_FR_03, O04_FR_04, O04_FR_05, O04_FR_06	
System under test	Charging Station	
Description	This test case describes how a CSO can request specific installed DisplayMessages configured via OCPP in a Charging Station. The Charging Station can remove messages when they are out-dated, or transactions have ended. It can be very useful for a CSO to be able to view to current list of messages, so the CSO knows which messages are (still) configured.	
Purpose	To verify if the Chargin Station is able to respond the specific priority messages requested by the CSMS according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: At least 1 message with *<Configured display_message_priority>* is configured	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **GetDisplayMessagesResponse**	**1.** The OCTT sends a **GetDisplayMessagesRequest** with **priority** *<Configured display_message_priority>* **requestId** *<Generated requestId>*
	3. The Charging Station sends a **NotifyDisplayMessagesRequest**	**4.** The OCTT responds with a **NotifyDisplayMessagesResponse** .
	Note(s): *- If **tbc** is True at Step 3 then step 3 and 4 will be repeated*	
Tool validations	* Step 2: Message **GetDisplayMessagesResponse** - **status** *Accepted* * Step 3: Message **NotifyDisplayMessagesRequest** - **requestId** *<Generated requestId>*	
	Post scenario validations: - All messages have been received	

Table 395. Test Case Id: TC_O_09_CS

Test case name	**Get a Specific Display Message - State**	
Test case Id	TC_O_09_CS	
Use case Id(s)	O04	
Requirement(s)	O04_FR_01, O04_FR_03, O04_FR_04, O04_FR_05, O04_FR_06	
System under test	Charging Station	
Description	This test case describes how a CSO can request specific installed DisplayMessages configured via OCPP in a Charging Station. The Charging Station can remove messages when they are out-dated, or transactions have ended. It can be very useful for a CSO to be able to view to current list of messages, so the CSO knows which messages are (still) configured.	
Purpose	To verify if the Chargin Station is able to respond the specific state messages requested by the CSMS according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: At least 1 message with *<Configured display_message_state>* is configured	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **GetDisplayMessagesResponse**	**1.** The OCTT sends a **GetDisplayMessagesRequest** with **state** *<Configured display_message_state>* **requestId** *<Generated requestId>*
	3. The Charging Station sends a **NotifyDisplayMessagesRequest**	**4.** The OCTT responds with a **NotifyDisplayMessagesResponse** .
	Note(s): *- If **tbc** is True at Step 3 then step 3 and 4 will be repeated*	
Tool validations	* Step 2: Message **GetDisplayMessagesResponse** - **status** *Accepted* * Step 3: Message **NotifyDisplayMessagesRequest** - **requestId** *<Generated requestId>*	
	Post scenario validations: - All messages have been received	

Table 396. Test Case Id: TC_O_10_CS

Test case name	**Set Display Message - Specific transaction - UnknownTransaction**	
Test case Id	TC_O_10_CS	
Use case Id(s)	O02	
Requirement(s)	O02_FR_01	
System under test	Charging Station	
Description	This test case describes how a CSO can set a message to be displayed on a Charging Station for a specific transaction. Depending on the given parameters the message shall be displayed a certain way on the Charging Station.	
Purpose	To verify if the Charging Station responds correctly according the mechanism as described in the OCPP specification when a display message request is received for an unknown specific transaction.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SetDisplayMessageResponse**	**1.** The OCTT sends a **SetDisplayMessageRequest** with **message.id** *<Generated displayMessageId>* AND **message.transactionId** *<Generated transactionId>* AND **message.priority** *<Configured Priority*
Tool validations	* Step 2: Message **SetDisplayMessageResponse** - **status** *UnknownTransaction*	
	Post scenario validations: - N/a	

Table 397. Test Case Id: TC_O_11_CS

Test case name	**Get a Specific Display Message - Unknown parameters**	
Test case Id	TC_O_11_CS	
Use case Id(s)	O04	
Requirement(s)	O04_FR_02	
System under test	Charging Station	
Description	This test case describes how a CSO can request specific installed DisplayMessages configured via OCPP in a Charging Station. The Charging Station can remove messages when they are out-dated, or transactions have ended. It can be very useful for a CSO to be able to view to current list of messages, so the CSO knows which messages are (still) configured.	
Purpose	To verify if the Chargin Station is able to respond correctly according to the mechanism as described in the OCPP specification when the specific id message requested by the CSMS is unknown.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: One display message with <Generated displayMessageId> is configured.	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **GetDisplayMessagesResponse**	**1.** The OCTT sends a **GetDisplayMessagesRequest** with **id** *<Other generated messageId>*
Tool validations	* Step 2: Message **GetDisplayMessagesResponse** - **status** *Unknown*	
	Post scenario validations: - N/a	

Table 398. Test Case Id: TC_O_12_CS

Test case name	**Set Display Message - Replace DisplayMessage**	
Test case Id	TC_O_12_CS	
Use case Id(s)	O06	
Requirement(s)	O06_FR_01	
System under test	Charging Station	
Description	This test case describes how a CSO can replace a DisplayMessage that is previously configured in a Charging Station. Replace the message content, but also all the given parameters with the new one.	
Purpose	To verify if the Chargin Station is able to replace a display message according to the DisplayMessage mechanism as described in the OCPP specification.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: Display message configured with <Generated displayMessageId>	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SetDisplayMessageResponse**	**1.** The OCTT sends a **SetDisplayMessageRequest** with **message.id** *<Generated displayMessageId from set Display Message>* **message.priority** *<Configured Priority*
	Note(s): - *The display message is replaced by a new one.*	
Tool validations	* Step 2: Message **SetDisplayMessageResponse** - **status** *Accepted*	
	Post scenario validations: - N/a	

Table 399. Test Case Id: TC_O_13_CS

Test case name	**Set Display Message - Display message at StartTime**	
Test case Id	TC_O_13_CS	
Use case Id(s)	O01	
Requirement(s)	O01_FR_06	
System under test	Charging Station	
Description	This test case describes how the CSMS can be requested to sent an SetDisplayMessageRequest to the charging station. Depending on the given parameters the message shall be displayed a certain way and at a certain moment on the Charging Station. These messages are displayed additionally on a Charging Station and are not part of the firmware.	
Purpose	To verify if the Charging Station is able to display additional messages with a certain start time according to the DisplayMessage mechanism as described in the OCPP specification.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SetDisplayMessageResponse**	**1.** The OCTT sends a **SetDisplayMessageRequest** with **message.id** *<Generated displayMessageId>* **message.priority** *<Configured Priority* **message.startDateTime** *<Current dateTime + 60 seconds>*
	4. The Charging Station responds with a **GetDisplayMessagesResponse**	**3.** The OCTT sends a **GetDisplayMessagesRequest** with **id** *<Generated displayMessageId>*
	5. The Charging Station sends a **NotifyDisplayMessagesRequest**	**6.** The OCTT responds with a **NotifyDisplayMessagesResponse** .
	Note(s): *- If **tbc** is True at Step 5 then step 5 and 6 will be repeated* *- Wait till 60 seconds are passed* *- The display message should be displayed after 60 seconds.*	
Tool validations	* Step 2: Message **SetDisplayMessageResponse** - **status** *Accepted* * Step 4: Message **GetDisplayMessagesResponse** - **status** *Accepted* * Step 5: Message **NotifyDisplayMessagesRequest** - **requestId** *<Generated requestId>* - **startDateTime** *<Should not be Omitted.>*	
	Post scenario validations: - N/a	

Table 400. Test Case Id: TC_O_14_CS

Test case name	**Set Display Message - Remove message after EndTime**	
Test case Id	TC_O_14_CS	
Use case Id(s)	O01	
Requirement(s)	O01_FR_07	
System under test	Charging Station	
Description	This test case describes how the CSMS can be requested to sent an SetDisplayMessageRequest to the charging station. Depending on the given parameters the message shall be displayed a certain way and at a certain moment on the Charging Station. These messages are displayed additionally on a Charging Station and are not part of the firmware.	
Purpose	To verify if the Charging Station is able to display additional messages with a certain end time according to the DisplayMessage mechanism as described in the OCPP specification.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SetDisplayMessageResponse**	**1.** The OCTT sends a **SetDisplayMessageRequest** with **message.id** *<Generated displayMessageId>* **message.priority** *<Configured Priority* **message.endDateTime** *<Current dateTime + 60 seconds>*
	4. The Charging Station responds with a **GetDisplayMessagesResponse**	**3.** The OCTT sends a **GetDisplayMessagesRequest** with **id** *<Generated displayMessageId>*
	5. The Charging Station sends a **NotifyDisplayMessagesRequest**	**6.** The OCTT responds with a **NotifyDisplayMessagesResponse** .
	<u>Note(s)</u>: *- If **tbc** is True at Step 5 then step 5 and 6 will be repeated* *- Wait till 60 seconds are passed* *- The display message is displayed and removed after 60 seconds.*	
	8. The Charging Station responds with a **GetDisplayMessagesResponse**	**7.** The OCTT sends a **GetDisplayMessagesRequest** with **id** *<Generated displayMessageId>* **requestId** *<Generated requestId>*
Tool validations	* Step 2: Message **SetDisplayMessageResponse** - **status** *Accepted* * Step 4: Message **GetDisplayMessagesResponse** - **status** *Accepted* * Step 5: Message **NotifyDisplayMessagesRequest** - **requestId** *<Generated requestId>* - **endDateTime** *<Should not be Omitted.>* * Step 8: Message **GetDisplayMessagesResponse** - **status** *Unknown*	
	Post scenario validations: - N/a	

Table 401. Test Case Id: TC_O_15_CS

Test case name	**Set Display Message - Language preference of the EV Driver**	
Test case Id	TC_O_15_CS	
Use case Id(s)	O01	
Requirement(s)	O01_FR_08	
System under test	Charging Station	
Description	This test case describes how the CSMS can be requested to sent an SetDisplayMessageRequest to the charging station. Depending on the given parameters the message shall be displayed a certain way and at a certain moment on the Charging Station. These messages are displayed additionally on a Charging Station and are not part of the firmware.	
Purpose	To verify if the Charging Station is able to set the preferred language according to the DisplayMessage mechanism as described in the OCPP specification.	
Prerequisite(s)	Charging station supports <Configured Language>	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State: **State is** *EVConnectedPreSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Present valid idToken which has a preferred language of <Configured language>*	
	1. The Charging Station sends an **AuthorizeRequest**	**2.** The OCTT responds with an **AuthorizeResponse** with **idTokenInfo.status** *Accepted* **idTokenInfo.language1** *<Configured language>*
	3. Execute **Reusable State** *EnergyTransferStarted*	
	5. The Charging Station responds with a **SetDisplayMessageResponse**	**4.** The OCTT sends a **SetDisplayMessageRequest** with **message.id** *<Generated displayMessageId>* **message.priority** *<Configured Priority* **message.message.content** *<Configured Message>*
	Note(s): *- The display message is displayed in the preferred language of the idToken as configured*	
Tool validations	* Step 1: Message **AuthorizeRequest** - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>* * Step 5: Message **SetDisplayMessageResponse** - **status** *Accepted*	
	Post scenario validations: - N/a	

Table 402. Test Case Id: TC_O_17_CS

Test case name	**Set Display Message - NotSupportedPriority**	
Test case Id	TC_O_17_CS	
Use case Id(s)	O01	
Requirement(s)	O01_FR_01, O02.FR.03	
System under test	Charging Station	
Description	This test case describes how the CSMS can be requested to sent an SetDisplayMessageRequest to the charging station. Depending on the given parameters the message shall be displayed a certain way and at a certain moment on the Charging Station. These messages are displayed additionally on a Charging Station and are not part of the firmware.	
Purpose	To verify if the Charging Station is able to respond correctly when the priority of the display messages is not supported according to the DisplayMessage mechanism as described in the OCPP specification.	
Prerequisite(s)	Charging station should not support all priorities described in the OCPP specification	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SetDisplayMessageResponse**	**1.** The OCTT sends a **SetDisplayMessageRequest** with **message.id** *<Generated displayMessageId>* **message.priority** *<Configured unsupported_display_message_priority>*
Tool validations	* Step 2: Message **SetDisplayMessageResponse** - **status** *NotSupportedPriority*	
	Post scenario validations: - N/a	

Table 403. Test Case Id: TC_O_18_CS

Test case name	**Set Display Message - NotSupportedState**	
Test case Id	TC_O_18_CS	
Use case Id(s)	O01	
Requirement(s)	O01_FR_02, O02.FR.04	
System under test	Charging Station	
Description	This test case describes how the CSMS can be requested to sent an SetDisplayMessageRequest to the charging station. Depending on the given parameters the message shall be displayed a certain way and at a certain moment on the Charging Station. These messages are displayed additionally on a Charging Station and are not part of the firmware.	
Purpose	To verify if the Charging Station is able to respond correclty when the state of the display messages is not supported according to the DisplayMessage mechanism as described in the OCPP specification.	
Prerequisite(s)	Charging station should not support all states described in the OCPP specification	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SetDisplayMessageResponse**	**1.** The OCTT sends a **SetDisplayMessageRequest** with **message.id** *<Generated displayMessageId>* **message.state** *<Configured unsupported_display_message_state>*
Tool validations	* Step 2: Message **SetDisplayMessageResponse** - **status** *NotSupportedState*	
	Post scenario validations: - N/a	

Table 404. Test Case Id: TC_O_19_CS

Test case name	**Set Display Message - NotSupportedMessageFormat**	
Test case Id	TC_O_19_CS	
Use case Id(s)	O01	
Requirement(s)	O01_FR_03, O02.FR.05	
System under test	Charging Station	
Description	This test case describes how the CSMS can be requested to sent an SetDisplayMessageRequest to the charging station. Depending on the given parameters the message shall be displayed a certain way and at a certain moment on the Charging Station. These messages are displayed additionally on a Charging Station and are not part of the firmware.	
Purpose	To verify if the Charging Station is able to respond correclty when the message format of the display messages is not supported according to the DisplayMessage mechanism as described in the OCPP specification.	
Prerequisite(s)	The Charging station does not support all formats described in the OCPP specification	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SetDisplayMessageResponse**	**1.** The OCTT sends a **SetDisplayMessageRequest** with **message.id** *<Generated displayMessageId>* Note(s): *The message is send in an unsupported format*
Tool validations	* Step 2: Message **SetDisplayMessageResponse** - **status** *NotSupportedMessageFormat*	
	Post scenario validations: - N/a	

Table 405. Test Case Id: TC_O_20_CS

Test case name	**Set Display Message - Persistent over reboot**	
Test case Id	TC_O_20_CS	
Use case Id(s)	O01	
Requirement(s)	O01_FR_10	
System under test	Charging Station	
Description	This test case describes how the CSMS can be requested to sent an SetDisplayMessageRequest to the charging station. Depending on the given parameters the message shall be displayed a certain way and at a certain moment on the Charging Station. These messages are displayed additionally on a Charging Station and are not part of the firmware.	
Purpose	To verify if the Charging Station is able to store display messages persistent over reboot according to the DisplayMessage mechanism as described in the OCPP specification.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SetDisplayMessageResponse**	**1.** The OCTT sends a **SetDisplayMessageRequest** with **message.id** *<Generated displayMessageId>* **message.priority** *<Configured Priority>*
	3. Execute **Reusable State** *Booted*	
	5. The Charging Station responds with a **GetDisplayMessagesResponse**	**4.** The OCTT sends a **GetDisplayMessagesRequest** with **id** *<Generated displayMessageId>* **requestId** *<Generated requestId>*
	6. The Charging Station sends a **NotifyDisplayMessagesRequest**	**7.** The OCTT responds with a **NotifyDisplayMessagesResponse** .
	Note(s): - *If **tbc** is True at Step 5 then step 5 and 6 will be repeated*	
Tool validations	* Step 2: Message **SetDisplayMessageResponse** - **status** *Accepted* * Step 5: Message **GetDisplayMessagesResponse** - **status** *Accepted* * Step 6: Message **NotifyDisplayMessagesRequest** - **requestId** *<RequestId sent in step 4>* - **id** *<Generated id>* - **priority** *<Configured Priority>* - **message.format** *<Configured format>* - **message.content** *<Configured content>*	
	Post scenario validations: - N/a	

Table 406. Test Case Id: TC_O_22_CS

Test case name	**Set Display Message - Multiple In front priority**	
Test case Id	TC_O_22_CS	
Use case Id(s)	O01	
Requirement(s)	O01_FR_14	
System under test	Charging Station	
Description	This test case describes how the CSMS can be requested to sent an SetDisplayMessageRequest to the charging station. Depending on the given parameters the message shall be displayed a certain way and at a certain moment on the Charging Station. These messages are displayed additionally on a Charging Station and are not part of the firmware.	
Purpose	To verify if the Charging Station is able to display multiple additional messages with a "InFront" priority according to the DisplayMessage mechanism as described in the OCPP specification.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SetDisplayMessageResponse**	**1.** The OCTT sends a **SetDisplayMessageRequest** with **message.id** *<Generated displayMessageId>* **message.priority** *InFront*
	4. The Charging Station responds with a **SetDisplayMessageResponse**	**3.** The OCTT sends a **SetDisplayMessageRequest** with **message.id** *<Generated displayMessage2Id>* **message.priority** *InFront*
	6. The Charging Station responds with a **GetDisplayMessagesResponse**	**5.** The OCTT sends a **GetDisplayMessagesRequest** with **id** *<Generated displayMessageId>* **requestId** *<Generated requestId>*
	7. The Charging Station sends a **NotifyDisplayMessagesRequest**	**8.** The OCTT responds with a **NotifyDisplayMessagesResponse** .
	10. The Charging Station responds with a **GetDisplayMessagesResponse**	**9.** The OCTT sends a **GetDisplayMessagesRequest** with **id** *<Generated displayMessage2Id>* **requestId** *<Generated requestId>*
	11. The Charging Station sends a **NotifyDisplayMessagesRequest**	**12.** The OCTT responds with a **NotifyDisplayMessagesResponse** .
	Note(s): *- If **tbc** is True at Step 7 then step 7 and 8 will be repeated* *- If **tbc** is True at Step 11 then step 11 and 12 will be repeated* *- The display messages are displayed as configured according the priority*	

Test case name	**Set Display Message - Multiple In front priority**
Tool validations	* Step 2: Message **SetDisplayMessageResponse** - **status** *Accepted* * Step 4: Message **SetDisplayMessageResponse** - **status** *Accepted* * Step 6: Message **GetDisplayMessagesResponse** - **status** *Accepted* * Step 7: Message **NotifyDisplayMessagesRequest** - **requestId** *<Generated requestId>* * Step 10: Message **GetDisplayMessagesResponse** - **status** *Accepted* * Step 11: Message **NotifyDisplayMessagesRequest** - **requestId** *<Generated requestId>*
	Post scenario validations: - N/a

Table 407. Test Case Id: TC_O_24_CS

Test case name	**Set Display Message - Second Alwaysfront priority**	
Test case Id	TC_O_24_CS	
Use case Id(s)	O01	
Requirement(s)	O01_FR_16	
System under test	Charging Station	
Description	This test case describes how the CSMS can be requested to sent an SetDisplayMessageRequest to the charging station. Depending on the given parameters the message shall be displayed a certain way and at a certain moment on the Charging Station. These messages are displayed additionally on a Charging Station and are not part of the firmware.	
Purpose	To verify if the Charging Station is able to display multiple additional messages with a "AlwaysFront" priority according to the DisplayMessage mechanism as described in the OCPP specification.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SetDisplayMessageResponse**	**1.** The OCTT sends a **SetDisplayMessageRequest** with **message.id** *<Generated displayMessageId>* **message.priority** *<Configured Priority>*
	4. The Charging Station responds with a **SetDisplayMessageResponse**	**3.** The OCTT sends a **SetDisplayMessageRequest** with **message.id** *<Configured displayMessage2Id>* **message.priority** *<Configured Priority>*
	6. The Charging Station responds with a **GetDisplayMessagesResponse**	**5.** The OCTT sends a **GetDisplayMessagesRequest** with **id** *<Generated displayMessageId>* **requestId** *<Generated requestId>*
	8. The Charging Station responds with a **GetDisplayMessagesResponse**	**7.** The OCTT sends a **GetDisplayMessagesRequest** with **id** *<Configured displayMessage2Id>*
	9. The Charging Station sends a **NotifyDisplayMessagesRequest**	**10.** The OCTT responds with a **NotifyDisplayMessagesResponse** .
	Note(s): *- If **tbc** is True at Step 7 then step 7 and 8 will be repeated* *- If **tbc** is True at Step 11 then step 11 and 12 will be repeated* *- The display messages are displayed as configured according the priority*	

Test case name	**Set Display Message - Second Alwaysfront priority**
Tool validations	* Step 2: Message **SetDisplayMessageResponse** - **status** *Accepted* * Step 4: Message **SetDisplayMessageResponse** - **status** *Accepted* * Step 6: Message **GetDisplayMessagesResponse** - **status** *Unknown* * Step 8: Message **GetDisplayMessagesResponse** - **status** *Accepted* * Step 9: Message **NotifyDisplayMessagesRequest** - **requestId** *<Generated requestId>*
	Post scenario validations: - N/a

Table 408. Test Case Id: TC_O_27_CS

Test case name	**Set Display Message - Specific transaction - Display message at StartTime**	
Test case Id	TC_O_27_CS	
Use case Id(s)	O02	
Requirement(s)	O02_FR_06	
System under test	Charging Station	
Description	This test case describes how a CSO can set a message to be displayed on a Charging Station for a specific transaction. Depending on the given parameters the message shall be displayed a certain way on the Charging Station.	
Purpose	To verify if the Charging Station is able to display the message with a certain start time correctly according the mechanism as described in the OCPP specification when a transaction is ongoing.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: **State** is *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SetDisplayMessageResponse**	**1.** The OCTT sends a **SetDisplayMessageRequest** with **message.id** *<Generated displayMessageId>* **message.priority** *<Configured Priority* **message.startDateTime** *<Current dateTime + 60 seconds>*
	Note(s): - *The display message is not yet displayed.* - *Waiting 60 seconds.* - *The display message is displayed after 60 seconds.*	
	3. Execute **Reusable State** *StopAuthorized*	
	4. Execute **Reusable State** *EVConnectedPostSession*	
	5. Execute **Reusable State** *EVDisconnected*	
	6. Execute **Reusable State** *ParkingBayUnoccupied*	
	Note(s): - *The display message is not displayed anymore*	
	8. The Charging Station responds with a **GetDisplayMessagesResponse**	**7.** The OCTT sends a **GetDisplayMessagesRequest** with **id** *<Generated displayMessageId>*
Tool validations	* Step 2: Message **SetDisplayMessageResponse** - **status** *Accepted* * Step 8: Message: **GetDisplayMessagesResponse** - **status** *Unknown*	
	Post scenario validations: - N/a	

Table 409. Test Case Id: TC_O_28_CS

Test case name	**Set Display Message - Specific transaction - Remove message after EndTime**	
Test case Id	TC_O_28_CS	
Use case Id(s)	O02	
Requirement(s)	O02_FR_07	
System under test	Charging Station	
Description	This test case describes how the CSMS can be requested to sent an SetDisplayMessageRequest to the charging station. Depending on the given parameters the message shall be displayed a certain way and at a certain moment on the Charging Station. These messages are displayed additionally on a Charging Station and are not part of the firmware.	
Purpose	To verify if the Charging Station is able to display additional messages with a certain end time for a specific transaction according to the DisplayMessage mechanism as described in the OCPP specification.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: **State** is *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SetDisplayMessageResponse**	**1.** The OCTT sends a **SetDisplayMessageRequest** with **message.id** *<Generated displayMessageId>* **message.priority** *<Configured Priority* **message.endDateTime** *<Current dateTime + 60 seconds>*
	Note(s): - *The display message should be displayed.* - *Waiting 60 seconds.* - *The display message is not being displayed anymore after 60 seconds.*	
	4. The Charging Station responds with a **GetDisplayMessagesResponse**	**3.** The OCTT sends a **GetDisplayMessagesRequest** with **id** *<Generated displayMessageId>*
Tool validations	* Step 2: Message **SetDisplayMessageResponse** - **status** *Accepted* * Step 4: Message **GetDisplayMessagesResponse** - **status** *Unknown*	
	Post scenario validations: - N/a	

Table 410. Test Case Id: TC_O_30_CS

Test case name	**Set Display Message - Specific transaction - Multiple In front priority**	
Test case Id	TC_O_30_CS	
Use case Id(s)	O02	
Requirement(s)	O02_FR_16	
System under test	Charging Station	
Description	This test case describes how a CSO can set a message to be displayed on a Charging Station for a specific transaction. Depending on the given parameters the message shall be displayed a certain way on the Charging Station.	
Purpose	To verify if the Charging Station is able to display multiple additional messages with a "InFront" priority for a specific transaction according to the DisplayMessage mechanism as described in the OCPP specification.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: **State** is *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SetDisplayMessageResponse**	**1.** The OCTT sends a **SetDisplayMessageRequest** with **message.id** *<Generated displayMessageId>* AND **message.transactionId** *<Received transactionId>* AND **message.priority** *InFront*
	4. The Charging Station responds with a **SetDisplayMessageResponse**	**3.** The OCTT sends a **SetDisplayMessageRequest** with **message.id** *<Generated displayMessageId2>* AND **message.transactionId** *<Received transactionId>* AND **message.priority** *InFront*
	Note(s): *- The display messages are displayed as configured*	
	6. The Charging Station responds with a **GetDisplayMessagesResponse**	**5.** The OCTT sends a **GetDisplayMessagesRequest** with **id** *<Generated displayMessageId>* **requestId** *<Generated requestId>*
	7. The Charging Station sends a **NotifyDisplayMessagesRequest**	**8.** The OCTT responds with a **NotifyDisplayMessagesResponse** .
	10. The Charging Station responds with a **GetDisplayMessagesResponse**	**9.** The OCTT sends a **GetDisplayMessagesRequest** with **id** *<Generated displayMessageId2>* **requestId** *<Generated requestId>*
	11. The Charging Station sends a **NotifyDisplayMessagesRequest**	**12.** The OCTT responds with a **NotifyDisplayMessagesResponse** .

Test case name	Set Display Message - Specific transaction - Multiple In front priority	
	13. Execute **Reusable State** *StopAuthorized*	
	14. Execute **Reusable State** *EVConnectedPostSession*	
	15. Execute **Reusable State** *EVDisconnected*	
	16. Execute **Reusable State** *ParkingBayUnoccupied*	
	Note(s): - *The display messages are not displayed anymore*	
	18. The Charging Station responds with a **GetDisplayMessagesResponse**	**17.** The OCTT sends a **GetDisplayMessagesRequest** with **id** *<Generated displayMessageId>*
	20. The Charging Station responds with a **GetDisplayMessagesResponse**	**19.** The OCTT sends a **GetDisplayMessagesRequest** with **id** *<Configured displayMessage2Id>*
Tool validations	* Step 2: Message **SetDisplayMessageResponse** - **status** *Accepted* * Step 4: Message **SetDisplayMessageResponse** - **status** *Accepted* * Step 6: Message: **GetDisplayMessagesResponse** - **status** *Accepted* * Step 7: Message: **NotifyDisplayMessagesRequest** - **requestId** *<Generated RequestId>* - **transactionId** *<Generated transactionId>* - **priority** *InFront* - **message.content** *<Configured message>* * Step 10: Message: **GetDisplayMessagesResponse** - **status** *Accepted* * Step 11: Message: **NotifyDisplayMessagesRequest** - **requestId** *<Generated RequestId>* - **transactionId** *<Generated transactionId>* - **priority** *InFront* - **message.content** *<Configured message with a " 2" extended to it.>* * Step 18: Message: **GetDisplayMessagesResponse** - **status** *Unknown* * Step 20: Message: **GetDisplayMessagesResponse** - **status** *Unknown*	
	Post scenario validations: - N/a	

Table 411. Test Case Id: TC_O_32_CS

Test case name	**Set Display Message - Specific transaction - Second Alwaysfront priority**	
Test case Id	TC_O_32_CS	
Use case Id(s)	O02	
Requirement(s)	O02_FR_18	
System under test	Charging Station	
Description	This test case describes how a CSO can set a message to be displayed on a Charging Station for a specific transaction. Depending on the given parameters the message shall be displayed a certain way on the Charging Station.	
Purpose	To verify if the Charging Station is able to display multiple additional messages with a "AlwaysFront" priority for a specific transaction according to the DisplayMessage mechanism as described in the OCPP specification.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: **State** is *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SetDisplayMessageResponse**	**1.** The OCTT sends a **SetDisplayMessageRequest** with **message.id** *<Generated displayMessageId>* **message.transactionId** *<Received transactionId>* AND **message.priority** *AlwaysFront*
	4. The Charging Station responds with a **SetDisplayMessageResponse**	**3.** The OCTT sends a **SetDisplayMessageRequest** with **message.id** *<Configured displayMessage2Id>* **message.transactionId** *<Received transactionId>* AND **message.priority** *AlwaysFront*
	6. The Charging Station responds with a **GetDisplayMessagesResponse**	**5.** The OCTT sends a **GetDisplayMessagesRequest** with **id** *<Generated displayMessageId>*
	8. The Charging Station responds with a **GetDisplayMessagesResponse**	**7.** The OCTT sends a **GetDisplayMessagesRequest** with **id** *<Generated displayMessageId2>* **requestId** *<Generated requestId>*
	9. The Charging Station sends a **NotifyDisplayMessagesRequest**	**10.** The OCTT responds with a **NotifyDisplayMessagesResponse** .
	11. Execute **Reusable State** *StopAuthorized*	
	12. Execute **Reusable State** *EVConnectedPostSession*	
	13. Execute **Reusable State** *EVDisconnected*	
	14. Execute **Reusable State** *ParkingBayUnoccupied*	
	Note(s): *- The display message is not displayed anymore*	
	16. The Charging Station responds with a **GetDisplayMessagesResponse**	**15.** The OCTT sends a **GetDisplayMessagesRequest** with **id** *<Generated displayMessageId2>*

Test case name	**Set Display Message - Specific transaction - Second Alwaysfront priority**
Tool validations	* Step 2: Message **SetDisplayMessageResponse** - **status** *Accepted* * Step 4: Message **SetDisplayMessageResponse** - **status** *Accepted* * Step 6: Message **GetDisplayMessagesResponse** - **status** *Unknown* * Step 8: Message **GetDisplayMessagesResponse** - **status** *Accepted* * Step 9: Message: **NotifyDisplayMessagesRequest** - **requestId** *<Generated RequestId>* - **transactionId** *<Generated transactionId>* - **priority** *AlwaysFront* - **message.content** *<Configured message with a " 2" extended to it.>* * Step 16: Message: **GetDisplayMessagesResponse** - **status** *Unknown*
	Post scenario validations: - N/a

Table 412. Test Case Id: TC_O_33_CS

Test case name	**Get a Specific Display Message - No DisplayMessages configured**	
Test case Id	TC_O_33_CS	
Use case Id(s)	O04	
Requirement(s)	O04_FR_07	
System under test	Charging Station	
Description	This test case describes how a CSO can request specific installed DisplayMessages configured via OCPP in a Charging Station. The Charging Station can remove messages when they are out-dated, or transactions have ended. It can be very useful for a CSO to be able to view to current list of messages, so the CSO knows which messages are (still) configured.	
Purpose	To verify if the Charging Station is able to respond correctly when a specific id message is requested by the CSMS but no messages are configured according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **GetDisplayMessagesResponse**	**1.** The OCTT sends a **GetDisplayMessagesRequest** with **id** *<Generated displayMessageId>*
Tool validations	* Step 2: Message **GetDisplayMessagesResponse** - **status** *Unknown*	
	Post scenario validations: - N/a	

Table 413. Test Case Id: TC_O_34_CS

Test case name	**Get a Specific Display Message - Known Id, but not matching State**	
Test case Id	TC_O_34_CS	
Use case Id(s)	O04	
Requirement(s)	O04_FR_02	
System under test	Charging Station	
Description	This test case describes how a CSO can request specific installed DisplayMessages configured via OCPP in a Charging Station. The Charging Station can remove messages when they are out-dated, or transactions have ended. It can be very useful for a CSO to be able to view to current list of messages, so the CSO knows which messages are (still) configured.	
Purpose	To verify if the Charging Station is able to respond correctly when a specific id message is requested by the CSMS but the requested **State is** different according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	Configured display message state 1, must be different than display message state 2.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: A display message is configured with <Generated displayMessageId> and <Configured display_message_state>	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **GetDisplayMessagesResponse**	**1.** The OCTT sends a **GetDisplayMessagesRequest** with **id** *<Generated displayMessageId>* AND **state** *<Configured display_message_2_state>*
Tool validations	* Step 2: Message **GetDisplayMessagesResponse** - **status** *Unknown*	
	Post scenario validations: - N/a	

Table 414. Test Case Id: TC_O_35_CS

Test case name	**Get a Specific Display Message - Known Id, but not matching Priority**	
Test case Id	TC_O_35_CS	
Use case Id(s)	O04	
Requirement(s)	O04_FR_02	
System under test	Charging Station	
Description	This test case describes how a CSO can request specific installed DisplayMessages configured via OCPP in a Charging Station. The Charging Station can remove messages when they are out-dated, or transactions have ended. It can be very useful for a CSO to be able to view to current list of messages, so the CSO knows which messages are (still) configured.	
Purpose	To verify if the Charging Station is able to respond correctly when a specific id message is requested by the CSMS but the requested priority is different according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	Configured display message priority 1, must be different than display message priority 2.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: A display message is configured with <Generated displayMessageId> and <Configured priority>	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **GetDisplayMessagesResponse**	**1.** The OCTT sends a **GetDisplayMessagesRequest** with **id** *<Generated displayMessageId>* AND **state** *<Configured display_message_2_priority>*
Tool validations	* Step 2: Message **GetDisplayMessagesResponse** - **status** *Unknown*	
	Post scenario validations: - N/a	

Table 415. Test Case Id: TC_O_36_CS

Test case name	**Set Display Message - State Charging**	
Test case Id	TC_O_36_CS	
Use case Id(s)	O01	
Requirement(s)	N/a	
System under test	Charging Station	
Description	This test case describes how the CSMS can be requested to sent an SetDisplayMessageRequest to the charging station. Depending on the given parameters the message shall be displayed a certain way and at a certain moment on the Charging Station. These messages are displayed additionally on a Charging Station and are not part of the firmware.	
Purpose	To verify if the Charging Station is able to display specific messages while the chargingState is Charging according to the DisplayMessage mechanism as described in the OCPP specification.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SetDisplayMessageResponse**	**1.** The OCTT sends a **SetDisplayMessageRequest** with **message.id** *<Generated displayMessageId>* **message.priority** *<Configured Priority* **message.state** *Charging*
	Note(s): *The display message should NOT be displayed.*	
	3. Execute **Reusable State** *ParkingBayOccupied*	
	4. Execute **Reusable State** *Authorized*	
	5. Execute **Reusable State** *EVConnectedPreSession*	
	6. Execute **Reusable State** *EnergyTransferStarted*	
	Note(s): *The display message should be displayed.*	
	7. Execute **Reusable State** *StopAuthorized*	
	8. Execute **Reusable State** *EVConnectedPostSession*	
	9. Execute **Reusable State** *EVDisconnected*	
	10. Execute **Reusable State** *ParkingBayUnoccupied*	
	Note(s): *The display message should NOT be displayed.*	
	12. The Charging Station responds with a **GetDisplayMessagesResponse**	**11.** The OCTT sends a **GetDisplayMessagesRequest** with **id** *<Generated displayMessageId>* **requestId** *<Generated requestId>*
	13. The Charging Station sends a **NotifyDisplayMessagesRequest**	**14.** The OCTT responds with a **NotifyDisplayMessagesResponse** .
	Note(s): *If **tbc** is True at Step 15 then step 15 and 16 will be repeated*	

Test case name	**Set Display Message - State Charging**
Tool validations	* Step 2: Message **SetDisplayMessageResponse** - **status** *Accepted* * Step 12: Message **GetDisplayMessagesResponse** - **status** *Accepted* * Step 13: Message **NotifyDisplayMessagesRequest** - **requestId** *<Generated requestId>* - **state** *Charging*
	Post scenario validations: - N/a

Table 416. Test Case Id: TC_O_37_CS

Test case name	**Set Display Message - State Idle**	
Test case Id	TC_O_37_CS	
Use case Id(s)	O01	
Requirement(s)	N/a	
System under test	Charging Station	
Description	This test case describes how the CSMS can be requested to sent an SetDisplayMessageRequest to the charging station. Depending on the given parameters the message shall be displayed a certain way and at a certain moment on the Charging Station. These messages are displayed additionally on a Charging Station and are not part of the firmware.	
Purpose	To verify if the Charging Station is able to display specific messages while the chargingState is Idle according to the DisplayMessage mechanism as described in the OCPP specification.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SetDisplayMessageResponse**	**1.** The OCTT sends a **SetDisplayMessageRequest** with **message.id** *<Generated displayMessageId>* **message.priority** *<Configured Priority* **message.state** *Idle*
	Note(s): *The display message should be displayed.*	
	3. Execute **Reusable State** *ParkingBayOccupied*	
	4. Execute **Reusable State** *Authorized*	
	5. Execute **Reusable State** *EVConnectedPreSession*	
	6. Execute **Reusable State** *EnergyTransferStarted*	
	Note(s): *The display message should NOT be displayed.*	
	7. Execute **Reusable State** *StopAuthorized*	
	8. Execute **Reusable State** *EVConnectedPostSession*	
	9. Execute **Reusable State** *EVDisconnected*	
	10. Execute **Reusable State** *ParkingBayUnoccupied*	
	Note(s): *The display message should be displayed.*	
	12. The Charging Station responds with a **GetDisplayMessagesResponse**	**11.** The OCTT sends a **GetDisplayMessagesRequest** with **id** *<Generated displayMessageId>* **requestId** *<Generated requestId>*
	13. The Charging Station sends a **NotifyDisplayMessagesRequest**	**14.** The OCTT responds with a **NotifyDisplayMessagesResponse** .
	Note(s): *If **tbc** is True at Step 13 then step 13 and 14 will be repeated*	

Test case name	**Set Display Message - State Idle**
Tool validations	* Step 2: Message **SetDisplayMessageResponse** - **status** *Accepted* * Step 12: Message **GetDisplayMessagesResponse** - **status** *Accepted* * Step 13: Message **NotifyDisplayMessagesRequest** - **requestId** *<Generated requestId>* - **state** *Idle*
	Post scenario validations: - N/a

Table 417. Test Case Id: TC_O_38_CS

Test case name	**Set Display Message - State Unavailable**	
Test case Id	TC_O_38_CS	
Use case Id(s)	O01	
Requirement(s)	N/a	
System under test	Charging Station	
Description	This test case describes how the CSMS can be requested to sent an SetDisplayMessageRequest to the charging station. Depending on the given parameters the message shall be displayed a certain way and at a certain moment on the Charging Station. These messages are displayed additionally on a Charging Station and are not part of the firmware.	
Purpose	To verify if the Charging Station is able to display specific messages while the chargingState is Unavailable according to the DisplayMessage mechanism as described in the OCPP specification.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SetDisplayMessageResponse**	**1.** The OCTT sends a **SetDisplayMessageRequest** with **message.id** *<Generated displayMessageId>* **message.priority** *<Configured Priority* **message.state** *Unavailable*
	Note(s): *The display message should NOT be displayed.*	
	3. Execute **Reusable State** *Unavailable*	
	Note(s): *The display message should be displayed.*	
	5. The Charging Station responds with a **ChangeAvailabilityResponse**	**4.** The OCTT sends a **ChangeAvailabilityRequest** with **operationalStatus** *Operative*
	6. The Charging Station notifies the CSMS about the current state of all its connectors (and optionally also the Charging Station itself and all EVSE).	**7.** The OCTT responds accordingly.
	Note(s): *The display message should NOT be displayed.*	
	9. The Charging Station responds with a **GetDisplayMessagesResponse**	**8.** The OCTT sends a **GetDisplayMessagesRequest** with **id** *<Generated displayMessageId>* **requestId** *<Generated requestId>*
	10. The Charging Station sends a **NotifyDisplayMessagesRequest**	**11.** The OCTT responds with a **NotifyDisplayMessagesResponse** .
	Note(s): *If **tbc** is True at Step 10 then step 10 and 11 will be repeated*	

Test case name	**Set Display Message - State Unavailable**
Tool validations	* Step 2: Message **SetDisplayMessageResponse** - **status** *Accepted* * Step 5: Message **ChangeAvailabilityResponse** - **status** *Accepted* * Step 6: Message: **StatusNotificationRequest** - **connectorStatus** *Available* Message: **NotifyEventRequest** - **eventData[0].trigger** *Delta* - **eventData[0].actualValue** *"Available"* - **eventData[0].component.name** *"ChargingStation" / EVSE / Connector* - **eventData[0].variable.name** *"AvailabilityState"* * Step 9: Message **GetDisplayMessagesResponse** - **status** *Accepted* * Step 10: Message **NotifyDisplayMessagesRequest** - **requestId** *<Generated requestId>* - **state** *Unavailable*
	Post scenario validations: - N/a

Table 418. Test Case Id: TC_O_39_CS

Test case name	**Set Display Message - State Faulted**	
Test case Id	TC_O_39_CS	
Use case Id(s)	O01	
Requirement(s)	N/a	
System under test	Charging Station	
Description	This test case describes how the CSMS can be requested to sent an SetDisplayMessageRequest to the charging station. Depending on the given parameters the message shall be displayed a certain way and at a certain moment on the Charging Station. These messages are displayed additionally on a Charging Station and are not part of the firmware.	
Purpose	To verify if the Charging Station is able to display specific messages while the chargingState is Faulted according to the DisplayMessage mechanism as described in the OCPP specification.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SetDisplayMessageResponse**	**1.** The OCTT sends a **SetDisplayMessageRequest** with **message.id** *<Generated displayMessageId>* **message.priority** *<Configured Priority* **message.state** *<Configured State>* **message.message** *Faulted*
	Note(s): *The display message should NOT be displayed.*	
	Manual Action: *Set the Charging Station to state Faulted.*	
	Note(s): *The display message should be displayed now.*	
	Manual Action: *Set the Charging Station back to state Available.*	
	Note(s): *The display message should NOT be displayed anymore.*	
	4. The Charging Station responds with a **GetDisplayMessagesResponse**	**3.** The OCTT sends a **GetDisplayMessagesRequest** with **id** *<Generated displayMessageId>* **requestId** *<Generated requestId>*
	5. The Charging Station sends a **NotifyDisplayMessagesRequest**	**6.** The OCTT responds with a **NotifyDisplayMessagesResponse** .
	Note(s): *If **tbc** is True at Step 5 then step 5 and 6 will be repeated*	
Tool validations	* Step 2: Message **SetDisplayMessageResponse** - **status** *Accepted* * Step 4: Message **GetDisplayMessagesResponse** - **status** *Accepted* * Step 5: Message **NotifyDisplayMessagesRequest** - **requestId** *<Generated requestId>* - **state** *Faulted*	
	Post scenario validations: - N/a	

2.17. P DataTransfer

Table 419. Test Case Id: TC_P_01_CS

Test case name	**Data Transfer to the Charging Station - Rejected / Unknown VendorId / Unknown MessageId**	
Test case Id	TC_P_01_CS	
Use case Id(s)	P01	
Requirement(s)	P01.FR.05, P01.FR.06	
System under test	Charging Station	
Description	The DataTransfer message to send information for functions that are not supported by OCPP.	
Purpose	To verify whether the Charging Station is able to handle receiving a DataTransferRequest, even if it does not support any vendor-specific implementations.	
Prerequisite(s)	The configured vendorId should not be implemented and the configured messageId should be unused.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **DataTransferResponse**	**1.** The OCTT sends a **DataTransferRequest** with **vendorId** *org.openchargealliance.octt* **messageId** *<Configured messageId>*
Tool validations	* Step 2: Message: **DataTransferResponse** - **status** must be *UnknownVendorId* OR *UnknownMessageId* OR *Rejected* (Rejected will also be allowed, because there are implementers that like to just reject the message when the Charging Station does not support any vendor-specific features.	
	Post scenario validations: N/a	

Table 420. Test Case Id: TC_P_03_CS

Test case name	**CustomData - Receive custom data**	
Test case Id	TC_P_03_CS	
Use case Id(s)	N/a	
Requirement(s)	N/a	
System under test	Charging Station	
Description	Checks if the CS is able to receive custom data.	
Purpose	To verify whether the CS is able to handle receiving custom data.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with **SetVariablesResponse**	**1.** OCTT sends **SetVariablesRequest** with: - **variable.name** = *"OfflineThreshold"* - **component.name** = *"OCPPCommCtrlr"* - **attributeValue** = *"200"* - **attributeType** is Actual
	4. The Charging Station responds with **GetVariablesResponse**	**3.** OCTT sends **GetVariablesRequest** with: - **variable.name** = *"OfflineThreshold"* - **component.name** = *"OCPPCommCtrlr"* - **attributeType** is Actual
Tool validations	* Step 2: Message: **SetVariablesResponse** - **setVariableResult[0].attributeStatus** *Accepted* * Step 4: Message: **GetVariablesResponse** - **getVariableResult[0].attributeStatus** *Accepted* - **getVariableResult[0].attributeType** *Actual* or omitted - **getVariableResult[0].attributeValue** *200*	
	Post scenario validations: - N/a	

Table 421. Test Case Id: TC_P_04_CS

Test case name	**Able to receive customData - ChargingProfile**	
Test case Id	TC_P_04_CS	
Use case Id(s)	N/a	
Requirement(s)	N/a	
System under test	Charging Station	
Description	Checks if the CS is able to receive custom data.	
Purpose	To verify whether the CS is able to handle receive custom data in smart charging profiles.	
Prerequisite(s)	The Charging Station supports Smart Charging	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SetChargingProfileResponse**	**1.** The OCTT sends a **SetChargingProfileRequest** with **evseId** *<Configured evseId>* **chargingProfile.id** *<Configured chargingProfileId>* **chargingProfile.chargingProfilePurpose** *TxDefaultProfile* **chargingProfile.customData** *<CustomData>* **chargingProfile.chargingSchedule.duration** *<Configured duration>* **chargingProfile.chargingSchedule.chargingRateUnit** *<Configured chargingRateUnit>* **chargingProfile.chargingSchedule.chargingSchedulePeriod.startPeriod** *0* **chargingProfile.chargingSchedule.chargingSchedulePeriod.limit** *if unit is A then 6(A) else 6000(W)* **chargingProfile.chargingSchedule.chargingSchedulePeriod.numberPhases** *<Configured numberPhases>* **chargingProfile.chargingSchedule.chargingSchedulePeriod.customData** *<CustomData>*
Tool validations	* Step 2: Message **SetChargingProfileResponse** - **status** *Accepted*	
	Post scenario validations: - N/a	

2.18. Reusable states

Testcases can refer to a reusable state at the before or main stage. The steps described at the reusable state will be executed and then it will return to the testcase that called the reusable state.

Table 422. Reusable State: Booting

State	**Booting**	
System under test	Charging Station	
Description	This state will prepare the Charging Station, so that it is still booting. The connection has not been setup yet.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **ResetResponse**	**1.** The OCTT sends a **ResetRequest** with **type** *Immediate*
Tool validations	* Step 2: Message: **ResetResponse** - **status** must be *Accepted*	
Post condition	**State** is *Booting*	

Table 423. Reusable State: Booted

State	**Booted**	
System under test	Charging Station	
Description	This state will reset or power cycle the Charging Station, depending on the testcase. The charging station ends in a state where it is booted back up and is in idle mode.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Scenario)	**Charging Station**	**CSMS**
	<u>Manual Action</u>: *Power cycle the Charging Station.* OR execute step 1 and 2, depending on the testcase.	
	2. The Charging Station responds with a **ResetResponse** with **status** *Accepted*	**1.** The OCTT sends a **ResetRequest**
	3. The Charging Station sends a **BootNotificationRequest**	**4.** The OCTT responds with a **BootNotificationResponse** with **status** *Accepted*
	5. The Charging Station notifies the CSMS about the current state of all connectors.	**6.** The OCTT responds accordingly.
	7 The Charging Station sends a **SecurityEventNotificationRequest**	**8** The OCTT responds with a **SecurityEventNotificationResponse**
Tool validations	* Step 2: Message: **ResetResponse** - **status** *Accepted* * Step 5: Message: **StatusNotificationRequest** - **connectorStatus** *Available* - **evseId** not *0* - **connectorId** not *0* Message: **NotifyEventRequest** - **eventData[0].trigger** *Delta* - **eventData[0].actualValue** *"Available"* - **eventData[0].component.name** *"Connector"* - **eventData[0].variable.name** *"AvailabilityState"* * Step 7: Message: **SecurityEventNotificationRequest** - **type** must be *StartupOfTheDevice* OR *ResetOrReboot*	
Post condition	**State** is *Booted*	

Table 424. Reusable State: Reserved

State	**Reserved**	
System under test	Charging Station	
Description	This state will prepare the Charging Station, so that one of its EVSE becomes reserved.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **ReserveNowResponse**	**1.** The OCTT sends a **ReserveNowRequest** with **evseId** is *<Specified evseId (Configured evseId as a default)>* **idToken.idToken** *<Specified valid_idtoken_idtoken (Configured idToken as a default)>* **idToken.type** *<Specified valid_idtoken_type>*
	3. The Charging Station notifies the CSMS about the status change of the connector. Note(s): *- The OCTT expects that the Charging Station sets the availabilityState of the EVSE and corresponding connectors to Reserved.* *- Reporting the AvailabilityState of the EVSE component itself is optional.*	**4.** The OCTT responds accordingly.
Tool validations	* Step 2: Message: **ReserveNowResponse** - **status** must be *Accepted* * Step 3: Message: **StatusNotificationRequest** - **evseId** not *0* - **connectorId** not *0* - **connectorStatus** must be *Reserved* Message: **NotifyEventRequest** - **eventData[0].trigger** must be *Delta* - **eventData[0].actualValue** must be *Reserved* - **eventData[0].component.name** must be *Connector* - **eventData[0].evse.id** not *0* - **eventData[0].evse.connectorId** not *0* - **eventData[0].variable.name** must be *AvailabilityState* (Optional) Message: **NotifyEventRequest** - **eventData[0].trigger** must be *Delta* - **eventData[0].actualValue** must be *Reserved* - **eventData[0].component.name** must be *EVSE* - **eventData[0].variable.name** must be *AvailabilityState*	
Post condition	**State** is *Reserved*	

Table 425. Reusable State: Unavailable

State	**Unavailable**	
System under test	Charging Station	
Description	This state will prepare the Charging Station, so that the Charging Station / EVSEs / connectors are set to AvailabilityState Unavailable.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **ChangeAvailabilityResponse**	**1.** The OCTT sends a **ChangeAvailabilityRequest** with **operationalStatus** *Inoperative* **evse.id** *<Specified evseId>* **evse.connectorId** *<Specified connectorId>*
	3. The Charging Station notifies the CSMS about the current state of all connectors belonging to the specified component(s).	**4.** The OCTT responds accordingly.
Tool validations	* Step 2: Message **ChangeAvailabilityResponse** - **status** *Accepted* * Step 3: Message: **StatusNotificationRequest** - **connectorStatus** *Unavailable* - **evseId** *<Specified evseId>* - **connectorId** *<Specified connectorId>* Message: **NotifyEventRequest** - **eventData[0].trigger** *Delta* - **eventData[0].actualValue** *"Unavailable"* - **eventData[0].component.name** *"ChargingStation" / EVSE / Connector* - **eventData[0].variable.name** *"AvailabilityState"*	
Post condition	**State** is *Reserved*	

Table 426. Reusable State: ParkingBayOccupied

State	**ParkingBayOccupied**	
System under test	Charging Station	
Description	This state will prepare the Charging Station, so that the EV entered the parking bay. The execution of this **State is** optional. Because there may not be a parking bay occupancy sensor OR the Charging Station is being tested with a test plug or EV simulator.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Scenario)	**Charging Station**	**CSMS**
	Manual Action: *Drive EV into parking bay.* Note(s): *- This **State is** optional (Even when TxStartPoint contains ParkingBayOccupancy).*	
	1. The Charging Station sends a **TransactionEventRequest** Note(s): *- This step needs to be executed when **TxStartPoint** contains ParkingBayOccupancy AND the EV entered the parking bay.*	**2.** The OCTT responds with a **TransactionEventResponse**
Tool validations	* Step 1: Message: **TransactionEventRequest** - **triggerReason** must be *EVDetected*	
Post condition	**State** is *ParkingBayOccupied*	

Table 427. Reusable State: EVConnectedPreSession

State	**EVConnectedPreSession**	
System under test	Charging Station	
Description	This state will prepare the Charging Station, so that the EV and EVSE are connected.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): If **State** is NOT *ParkingBayOccupied* then execute **Reusable State** *ParkingBayOccupied*	
Main (Scenario)	**Charging Station**	**CSMS**
	Manual Action: *Connect the EV and EVSE.*	
	1. The Charging Station notifies the CSMS about the status change of the connector.	**2.** The OCTT responds accordingly.
	3. The Charging Station sends a **TransactionEventRequest** Note(s): *- This step needs to be executed when* ***TxStartPoint*** *contains EVConnected OR the transaction already started. So in the case* ***TxStartPoint*** *contains ParkingBayOccupancy OR Authorized*	**4.** The OCTT responds with a **TransactionEventResponse**
Tool validations	* Step 1: Message: **StatusNotificationRequest** - **evseId** *<configured evseId>* - **connectorId** *<configured connectorId>* - **connectorStatus** must be *Occupied* Message: **NotifyEventRequest** - **eventData[0].trigger** must be *Delta* - **eventData[0].actualValue** must be *Occupied* - **eventData[0].component.name** must be *Connector* - **eventData[0].variable.name** must be *AvailabilityState* - **evse.id** *<configured evseId>* - **connector.id** *<configured connectorId>* * Step 3: Message: **TransactionEventRequest** - **eventType** *started if* ***TxStartPoint*** *is EVConnected or PowerPathClosed and* ***State*** *is Authorized, else updated* - **triggerReason** must be *CablePluggedIn* or *ChargingStateChanged* or *RemoteStart* - **transactionInfo.chargingState** must be *EVConnected or SuspendedEVSE or Charging if* ***State*** *is Authorized* - **evse.id** *<configured evseId>* - **connector.id** *<configured connectorId>*	
Post condition	**State** is *EVConnectedPreSession*	

Table 428. Reusable State: Authorized

State	**Authorized**	
System under test	Charging Station	
Description	This state will prepare the Charging Station, so that the transaction is authorized. This can be done in two ways (The default way is configurable at OCTT. This will be used when the calling testcase does not define which one to use.): A. Using local authorization B. Using a **RequestStartTransactionRequest**	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): If **State** is NOT *ParkingBayOccupied* OR *EVConnectedPreSession*, then execute **Reusable State** *ParkingBayOccupied*	
Main A (Scenario)	**Charging Station**	**CSMS**
	Manual Action: *Present idToken.*	
	1. The Charging Station sends an **AuthorizeRequest** Note(s): *- This step needs to be executed, unless (**AuthEnabled** is implemented with mutability ReadOnly AND the value is set to false) OR a start button as described at Use case C02 is used (This must be configured at the OCTT) OR the **idToken** is cached. In case the **idToken** is used for a reservation, sending the **AuthorizeRequest** message is optional.*	**2.** The OCTT responds with an **AuthorizeResponse** with **idTokenInfo.status** *Accepted*
	3. The Charging Station sends a **TransactionEventRequest** Note(s): *- This step needs to be executed when **TxStartPoint** contains Authorized OR the transaction already started. So in the case **TxStartPoint** contains ParkingBayOccupancy or (EVConnected, in the case this testcase was initiated from state EVConnectedPreSession.)*	**4.** The OCTT responds with a **TransactionEventResponse** Note(s): *- The first TransactionEventRequest sent after authorization contains the idToken field, unless a Start button was used to start the transaction. In case there is an idToken used, the TransactionEventResponse of this request message contains **idTokenInfo** with **status** Accepted*
Tool validations	* Step 1: Message: **AuthorizeRequest** - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>* * Step 3: Message: **TransactionEventRequest** - **triggerReason** must be *Authorized* - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>*	

State	**Authorized**	
Main B (Scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **RequestStartTransactionResponse**	**1.** The OCTT sends a **RequestStartTransactionRequest** with **idToken.idToken** *<Configured valid_idtoken_idtoken>* **idToken.type** *<Configured valid_idtoken_type>* **evseId** *<Configured evseId>*
	3. The Charging Station sends an **AuthorizeRequest** Note(s): *- This step needs to be executed when **AuthCtrlr.AuthorizeRemoteStart** is true, unless (**AuthEnabled** is implemented with mutability ReadOnly AND the value is set to false) OR the **idToken** is cached.* *In case the **idToken** is used for a reservation, sending the **AuthorizeRequest** message is optional.*	**4.** The OCTT responds with an **AuthorizeResponse** with **idTokenInfo.status** *Accepted*
	5. The Charging Station sends a **StatusNotificationRequest** with: **connectorStatus** *Occupied*	**6.** The OCTT responds with a **StatusNotificationResponse**
	7. The Charging Station sends a **TransactionEventRequest** Note(s): *- This step needs to be executed when **TxStartPoint** contains Authorized OR the transaction already started. So in the case **TxStartPoint** contains ParkingBayOccupancy or (EVConnected, in the case this testcase was initiated from state EVConnectedPreSession.)*	**8.** The OCTT responds with a **TransactionEventResponse** Note(s): *- The first TransactionEventRequest sent after authorization contains the idToken field. The TransactionEventResponse of this request message contains **idTokenInfo** with **status** Accepted*
Tool validations	* Step 2: Message: **RequestStartTransactionResponse** - **status** must be *Accepted* If the transaction has already been started, so if TxStartPoint contains *ParkingBayOccupancy* OR (<Configured TxStartPoint> contains *EVConnected* AND State pre reusable state execution was *EVConnectedPreSession*) then - **transactionId** must be *<Provided transactionId in first TransactionEventRequest>* * Step 3: Message: **AuthorizeRequest** - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>* * Step 5: Message: **TransactionEventRequest** - **eventType** *Started if **TxStartPoint** is Authorized or PowerPathClosed and and **State** is EVConnectedPreSession, else updated* - **triggerReason** must be *RemoteStart* - **transactionInfo.remoteStartId** must be present. - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>*	
Post condition	**State** is *Authorized*	

Table 429. Reusable State: Authorized15118

State	**Authorized15118**	
System under test	Charging Station	
Description	This state will prepare the Charging Station, so that the transaction is authorized. This can be done in two ways based on the value of the *Authorization Method* config varaible: A. *EIM*, using a valid id token B. *PnC*, plug and charge	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Scenario)	**Charging Station**	**CSMS**
	Manual Action: *Present idToken* if configured authorization method is *EIM*	
	1. The Charging Station sends an **AuthorizeRequest** Note(s): *-The test case should be robust enough to also handle a* **GetCertificateStatusRequest** *and then expect the* **AuthorizeRequest**.	**2.** The OCTT responds with an **AuthorizeResponse** with **idTokenInfo.status** *Accepted*

Table 430. Reusable State: EnergyTransferStarted

State	**EnergyTransferStarted**	
System under test	Charging Station	
Description	This state will prepare the Charging Station, so that the Charging Station is transferring energy between the EV and EVSE.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): If **State** is NOT *Authorized* then execute **Reusable State** *Authorized* If **EVConnected** is *true*, then proceed to part 2 Else proceed to part 1.	
Main (Part 1) (Scenario)	**Charging Station**	**CSMS**
	Manual Action: *Connect the EV and EVSE.*	
	1. The Charging Station notifies the CSMS about the status change of the connector.	**2.** The OCTT responds accordingly.
	3. The Charging Station sends a **TransactionEventRequest** Note(s): *- This step needs to be executed when **TxStartPoint** contains EVConnected OR the transaction already started. So in the case **TxStartPoint** contains ParkingBayOccupancy OR Authorized*	**4.** The OCTT responds with a **TransactionEventResponse**
Tool validations	* Step 1: Message: **StatusNotificationRequest** - **connectorStatus** must be *Occupied* Message: **NotifyEventRequest** - **eventData[0].trigger** must be *Delta* - **eventData[0].actualValue** must be *Occupied* - **eventData[0].component.name** must be *Connector* - **eventData[0].variable.name** must be *AvailabilityState* * Step 3: Message: **TransactionEventRequest** - **triggerReason** must be *CablePluggedIn* - **transactionInfo.chargingState** must be *EVConnected*	

State	**EnergyTransferStarted**	
Main (Part 2) (Scenario)	**Charging Station**	**CSMS**
	5. The Charging Station sends a **TransactionEventRequest** Note(s): *- This step only needs to be executed when **TxStartPoint** contains DataSigned AND the transaction was not already started. So in the case **TxStartPoint** also contains ParkingBayOccupancy OR EVConnected OR Authorized*	**6.** The OCTT responds with a **TransactionEventResponse**
	7. The Charging Station sends a **TransactionEventRequest** Note(s): *- This step only needs to be executed when **TxStartPoint** contains PowerPathClosed AND the transaction was not already started. So in the case **TxStartPoint** also contains ParkingBayOccupancy OR EVConnected OR Authorized OR DataSigned*	**8.** The OCTT responds with a **TransactionEventResponse**
	9. The Charging Station sends a **TransactionEventRequest** Note(s): *- This step needs to be executed when **TxStartPoint** contains EnergyTransfer OR the transaction already started. So in the case **TxStartPoint** contains ParkingBayOccupancy OR EVConnected OR Authorized OR DataSigned OR PowerPathClosed*	**10.** The OCTT responds with a **TransactionEventResponse**
Tool validations	* Step 5: Message: **TransactionEventRequest** - **triggerReason** must be *SignedDataReceived* * Step 7: Message: **TransactionEventRequest** - **triggerReason** must be *ChargingStateChanged* - **transactionInfo.chargingState** must be *SuspendedEVSE* * Step 9: Message: **TransactionEventRequest** - **triggerReason** must be *ChargingStateChanged* - **transactionInfo.chargingState** must be *Charging*	
Post condition	**State** is *EnergyTransferStarted* **EVConnected** is *true*	

Table 431. Reusable State: EnergyTransferSuspended

State	**EnergyTransferSuspended**	
System under test	Charging Station	
Description	This state will prepare the Charging Station, so that it is in a state where the energy transfer is suspended by the EV.	
Prerequisite	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): If **State** is NOT *EnergyTransferStarted* then execute **Reusable State** *EnergyTransferStarted*	
Main (Scenario)	**Charging Station**	**CSMS**
	Notes(s): *The tool will wait for <Configured Transaction Duration> seconds*	
	Manual Action: *The EV suspends the energy transfer.*	
	1. The Charging Station sends a **TransactionEventRequest** Note(s): - *This step needs to be executed unless the transaction was already stopped. So in the case* ***TxStopPoint*** *contains _EnergyTransfer*	**2.** The OCTT responds with a **TransactionEventResponse**
Tool validations	* 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*	
Post condition	**State** is *EnergyTransferSuspended*	

Table 432. Reusable State: StopAuthorized

State	StopAuthorized	
System under test	Charging Station	
Description	This state will prepare the Charging Station, so that it is in a state where the charging session is authorized to stop. This can be done in two ways (Configurable at OCTT): A. Using local authorization B. Using a **RequestStopTransactionRequest**	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): If **State** is NOT *EnergyTransferStarted* then execute **Reusable State** *EnergyTransferStarted* Note: The OCTT will wait a number of seconds equal to the configured <*TransactionDuration*>, before proceeding to the Main stage.	
Main A (Scenario)	**Charging Station**	**CSMS**
	Notes(s): *The tool will wait for <Configured Transaction Duration> seconds*	
	Manual Action: *Present the same idToken as used to start the transaction.*	
	1. The Charging Station sends a **TransactionEventRequest**	**2.** The OCTT responds with a **TransactionEventResponse** With **idTokenInfo.status** is *Accepted*
	Note(s): This step is optional	
	3. The Charging Station sends a **TransactionEventRequest**	**4.** The OCTT responds with a **TransactionEventResponse** With **idTokenInfo.status** is *Accepted*
Tool validations	* Step 1: Message: **TransactionEventRequest** - **triggerReason** must be *StopAuthorized* - **idToken** *omit* OR - **idToken.idToken** *<Configured valid_idtoken_idtoken>* AND - **idToken.type** *<Configured valid_idtoken_type>* * Step 3: Message: **TransactionEventRequest** - **triggerReason** must be *ChargingStateChanged* - **transactionInfo.chargingState** must be *EVConnected* - **eventType** must be *Ended* - **transactionInfo.stoppedReason** must be *Local* or omitted	
Main B (Scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **RequestStopTransactionResponse**	**1.** The OCTT sends a **RequestStopTransactionRequest** with **transactionId** *<transactionId provided by the Charging Station in **TransactionEventRequest**>*
	3. The Charging Station sends a **TransactionEventRequest**	**4.** The OCTT responds with a **TransactionEventResponse**
Tool validations	* Step 2: Message: **RequestStopTransactionResponse** - **status** must be *Accepted* * Step 3: Message: **TransactionEventRequest** - **triggerReason** must be *RemoteStop*	

State	**StopAuthorized**
Post condition	**State** is *StopAuthorized*

Table 433. Reusable State: EVConnectedPostSession

State	**EVConnectedPostSession**	
System under test	Charging Station	
Description	This state will prepare the Charging Station, so that energy transfer has been stopped and the transaction is NOT authorized to resume energy transfer without re-authorization.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): If **State** is NOT *StopAuthorized* then execute **Reusable State** *StopAuthorized*	
Main (Scenario)	**Charging Station**	**CSMS**
	1. The Charging Station sends a **TransactionEventRequest** Note(s): *- This step needs to be executed when the transaction has NOT been ended already. So in the case* ***TxStopPoint*** *contains Authorized OR PowerPathClosed*	**2.** The OCTT responds with a **TransactionEventResponse**
	3. The Charging Station sends a **TransactionEventRequest** Note(s): *- This step only needs to be executed when* ***TxStopPoint*** *contains DataSigned AND the transaction has NOT been ended already. So in the case* ***TxStopPoint*** *contains Authorized OR EnergyTransfer OR PowerPathClosed*	**4.** The OCTT responds with a **TransactionEventResponse**
Tool validations	* Step 1: Message: **TransactionEventRequest** - **triggerReason** must be *ChargingStateChanged* - **transactionInfo.chargingState** must be *EVConnected* * Step 3: Message: **TransactionEventRequest** - **triggerReason** must be *SignedDataReceived*	
Post condition	**State** is *EVConnectedPostSession*	

Table 434. Reusable State: EVDisconnected

State	**EVDisconnected**	
System under test	Charging Station	
Description	This state will prepare the Charging Station, so that the EV and EVSE are disconnected, after the charging session is authorized to stop.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): If **State** is NOT *EVConnectedPostSession* then execute **Reusable State** *EVConnectedPostSession*	
Main (Scenario)	**Charging Station**	**CSMS**
	Manual Action: *Disconnect the EV and EVSE.*	
	1. The Charging Station notifies the CSMS about the status change of the connector.	**2.** The OCTT responds accordingly.
	3. The Charging Station sends a **TransactionEventRequest** Note(s): *- This step needs to be executed when the transaction has NOT been ended already. So in the case **TxStopPoint** contains Authorized OR EnergyTransfer OR PowerPathClosed OR DataSigned*	**4.** The OCTT responds with a **TransactionEventResponse**
Tool validations	* Step 1: Message: **StatusNotificationRequest** - **connectorStatus** must be *Available* - **evseId** must be *<configured evseId>* - **connectorId** must be *<configured connectorId>* Message: **NotifyEventRequest** - **eventData[0].trigger** must be *Delta* - **eventData[0].actualValue** must be *Available* - **eventData[0].component.name** must be *Connector* - **eventData[0].variable.name** must be *AvailabilityState* - **eventData[0].component.evse.id** must be *<configured evseId>* - **eventData[0].component.evse.connectorId** must be *<configured connectorId>* * Step 3: Message: **TransactionEventRequest** - **triggerReason** must be *EVCommunicationLost* - **transactionInfo.chargingState** must be *Idle*	
Post condition	**State** is *EVDisconnected*	

Table 435. Reusable State: ParkingBayUnoccupied

State	**ParkingBayUnoccupied**	
System under test	Charging Station	
Description	This state will prepare the Charging Station, so that the EV left the parking bay, after a charging session has taken place.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): If **State** is NOT *EVDisconnected* then execute **Reusable State** *EVDisconnected*	
Main (Scenario)	**Charging Station**	**CSMS**
	Manual Action: *Drive EV out of parking bay.*	
	1. The Charging Station sends a **TransactionEventRequest** Note(s): *- This step needs to be executed when **TxStopPoint** contains ParkingBayOccupancy AND the transaction has NOT been ended already. So in the case **TxStopPoint** contains Authorized OR EnergyTransfer OR PowerPathClosed OR DataSigned OR EVConnected.*	**2.** The OCTT responds with a **TransactionEventResponse**
Tool validations	* Step 1: Message: **TransactionEventRequest** - **triggerReason** must be *EVDeparted* - If the OCTT is configured to stop transactions using a RequestStopTransactionRequest message then **transactionInfo.stoppedReason** must be *Remote* Else **transactionInfo.stoppedReason** must be *Local* - **eventType** must be *Ended*	
Post condition	**State** is *ParkingBayUnoccupied*	

Table 436. Reusable State: StartOfflineTransaction

State	**StartOfflineTransaction**	
System under test	Charging Station	
Description	This state will start a transaction while the Charging Station is offline.	
Prerequisite		
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Scenario)	**Charging Station**	**CSMS**
	1. *The OCTT closes the WebSocket connection AND does not accept a reconnect.*	
	Manual Action: *Drive EV into parking bay.*	
	Manual Action: *Present idToken.*	
	Manual Action: *Connect the EV and EVSE.*	
	2. *The OCTT accepts reconnection attempt from the Charging Station.*	
Tool validations	N/a	
Post condition	N/a	

Table 437. Reusable State: RenegotiateChargingLimits

State	**RenegotiateChargingLimits**	
System under test	Charging Station	
Description	…	
Prerequisite		
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Scenario)	**Charging Station**	**CSMS**
	Manual Action: *Renegotiate EV Charging Limits*	
	1. The Charging Station sends a **NotifyEVChargingNeedsRequest** with **evseId** *<Configured evseId>*	**2.** The OCTT responds with a **NotifyEVChargingNeedsResponse** with **status** *Accepted*
	4. The Charging Station responds with a **SetChargingProfileResponse** with **status** *Accepted*	**3.** The OCTT sends a **SetChargingProfileRequest** with **chargingProfile**: **.chargingProfilePurpose** *TxProfile* **.transactionId** *<Provided transactionId from before>* **chargingProfile.chargingSchedule[0]**: **.duration** *300* **.chargingRateUnit** *<Configured chargingRateUnit>* *Note: If <Configured chargingRateUnit> is W, then the **limit** field will be multiplied by 1000.* **.chargingSchedulePeriod[0].startPeriod** *0* If <Configured chargingRateUnit> is W: **.chargingSchedulePeriod[0].limit** *10000* else: **.chargingSchedulePeriod[0].limit** *10*
	5. The Charging Station sends a **NotifyEVChargingScheduleRequest** with **evseId** *<Configured evseId>*	**6.** The OCTT responds with a **NotifyEVChargingScheduleResponse** with **status** *Accepted*
	Note: *Steps 5 and 6 are optional. The Charging Station will only send a NotifyEVChargingScheduleRequest when the EV returns a charging profile.*	
	7. The Charging Station sends a **TransactionEventRequest**	**8.** The OCTT responds with a **TransactionEventResponse**
	Note: Steps 7 and 8 are optional, but can also repeat until chargingState is Charging.	

State	**RenegotiateChargingLimits**
Tool validations	* Step 1: Message: **NotifyEVChargingNeedsRequest**) - **evseId** *<Configured evseId>* - if **chargingNeeds.requestedEnergyTransfer** is *DC*: - **chargingNeeds.dcChargingParameters** should not be omitted - else: - **chargingNeeds.acChargingParameters** should not be omitted * Step 4: Message: **SetChargingProfileResponse**) - **status** *Accepted* * Step 5: Message: **NotifyEVChargingScheduleRequest**) - **evseId** *<Configured evseId>* * Step 7: Message: **TransactionEventRequest** - **triggerReason** must be *ChargingStateChanged* - **transactionInfo.chargingState** must be *Charging*
Post condition	N/a

Table 438. Reusable State: GetInstalledCertificates

State	**GetInstalledCertificates**	
System under test	Charging Station	
Description	The hashData from installed certificates of the specified type will be retrieved from the Charging Station	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **GetInstalledCertificateIdsResponse**	**1.** The OCTT sends a **GetInstalledCertificateIdsRequest** With **certificateType** is *<Specified certificateType>*
Tool validations	* Step 2: Message: **GetInstalledCertificateIdsResponse** - **status** must be *Accepted* - **certificateHashDataChain** must contain an entry with following values: *Note: Order does not matter.* - **certificateHashDataChain[0].certificateType** is *<Specified certificateType>* - **certificateHashDataChain[0].certificateHashData** contains *<HashData from the configured certificate of the specified certificateType>*	
Post condition	Certificate of the specified certificateType is retrieved from the Charging Station.	

Table 439. 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 440. 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	

2.19. Memory states

Table 441. Memory State: TransactionEventsInQueueEnded

State	**TransactionEventsInQueueEnded**	
System under test	Charging Station	
Description	This state will prepare the Charging Station, so that there will be TransactionEventRequests stored in its queue from an ended Transaction.	
Before (Preparations)	**Configuration State:** **OfflineTxForUnknownIdEnabled** is *true* (If implemented)	
	Memory State: *IdTokenCached* for <*Configured valid IdToken fields*> (If implemented) *IdTokenLocalAuthList* for <*Configured valid IdToken fields*> (If implemented)	
	Reusable State(s): N/a	
Main (Scenario)	**Charging Station**	**CSMS**
	1. *The OCTT closes the WebSocket connection AND does not accept a reconnect.*	
	Manual Action: *Drive EV into parking bay.*	
	Manual Action: *Connect the EV and EVSE.*	
	Manual Action: *Present idToken.*	
	Manual Action: *Present the same idToken as used to start the transaction.*	
	Manual Action: *Disconnect the EV and EVSE.*	
	Manual Action: *Drive EV out of parking bay.*	
	2. *The OCTT accepts reconnection attempt from the Charging Station.*	
Tool validations	N/a	
Post condition	TransactionEventRequest messages are stored in the queue of the Charging Station.	

Table 442. Memory State: CertificateInstalled

State	**CertificateInstalled**	
System under test	Charging Station	
Description	A pre configured certificate of the specified certificateType will be installed.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **InstallCertificateResponse**	**1.** The OCTT sends a **InstallCertificateRequest** with **certificateType** is *<Specified certificateType>* **certificate** is *<Corresponding certificate>*
Tool validations	* Step 2: Message: **InstallCertificateResponse** - **status** must be *Accepted*	
Post condition	Certificate of the specified certificateType is stored at the Charging Station.	

Table 443. Memory State: IdTokenCached

State	**IdTokenCached**	
System under test	Charging Station	
Description	An idToken is stored in the Authorization Cache of the Charging Station.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *ParkingBayoccupied*	
	2. Execute **Reusable State** *Authorized*	
Main A (Scenario)	**Charging Station**	**CSMS**
	Note(s): In case idToken is Accepted	
	3. Execute **Reusable State** *EVConnectedPreSession*	
	4. Execute **Reusable State** *ParkingBayUnoccupied*	
Tool validations	N/a	
Main B (Scenario)	**Charging Station**	**CSMS**
	Note(s): In case idToken is not Accepted	
	3. The Charging Station sends a **TransactionEventRequest**	**4.** The OCTT responds with a **TransactionEventResponse**
	Note(s): *Steps 3 and 4 are optional depending on the TxStartPoint*	
	5. Execute **Reusable State** *ParkingBayUnoccupied*	
Tool validations	* Step 3: Message: **TransactionEventRequest** - **triggerReason** must be *EVConnectionLost* - **transactionInfo.chargingState** must be *Idle*	
Post condition	N/a	

Table 444. Memory State: IdTokenLocalAuthList

State	**IdTokenLocalAuthList**	
System under test	Charging Station	
Description	An valid idToken is stored in the Local Authorization List of the Charging Station.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SendLocalListResponse**	**1.** The OCTT sends a **SendLocalListRequest** with **updateType** *Full* **localAuthorizationList[0].idToken.idToken** *<Configured valid_idtoken_idtoken>* **localAuthorizationList[0].idToken.type** *<Configured valid_idtoken_type>*
Tool validations	* Step 2: (Message: **SendLocalListResponse**) **status** is *Accepted*	
Post condition	N/a	

Table 445. Memory State: SetChargingProfile

State	**SetChargingProfile**	
System under test	Charging Station	
Description	This will store a Charging Profile at the Charging Station.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **SetChargingProfileResponse**	**1.** The OCTT sends a **SetChargingProfileRequest** with **chargingProfile** *<Provided chargingProfile>*
Tool validations	* Step 2: (Message: **SetChargingProfileResponse**) **status** is *Accepted*	
Post condition	N/a	

Table 446. Memory State: RenewChargingStationCertificate

State	**RenewChargingStationCertificate**	
System under test	Charging Station	
Description	The ChargingStationCertificate is renewed using A02/A03	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **TriggerMessageResponse**	**1.** The OCTT sends a **TriggerMessageRequest** With **requestedMessage** *SignChargingStationCertificate*
	3 The Charging Station sends a **SignCertificateRequest**	**4.** The OCTT responds with a **SignCertificateResponse** With **status** *Accepted*
	6. The Charging Station responds with a **CertificateSignedResponse**	**5.** The OCTT sends a **CertificateSignedRequest** With **certificateChain** *<Certificate generated from the received CSR from step 3 and signed by the provided CSMS Root certificate>* **certificateType** *ChargingStationCertificate*
Tool validations	* Step 2: Message: **TriggerMessageResponse** - **status** must be *Accepted* * Step 3: Message: **SignCertificateRequest** - **csr** must contain *<An CSR that meets the following requirements:* *When using RSA or DSA the key must be at least 2048 bits long.* *and when using elliptic curve cryptography the key must be at least 224 bits long.* *The received CSR must be transmitted as described in RFC 2986 and then encoded in Privacy-Enhanced Mail (PEM) format.>* * Step 6: Message: **CertificateSignedResponse** - **status** must be *Accepted*	
	Post scenario validations: N/a	

Table 447. Memory State: RenewV2GChargingStationCertificate

State	RenewV2GChargingStationCertificate	
System under test	Charging Station	
Description	The V2G ChargingStationCertificate is renewed using A02/A03	
Before (Preparations)	**Configuration State:** **ISO15118Ctrlr.V2GCertificateInstallationEnabled** is *true* if implementated **ISO15118Ctrlr.CountryName** is *NL* if implemented **ISO15118Ctrlr.OrganizationName** is configured vendorId if implemented OCTT will check all configured **ISO15118Ctrlr.SeccId's** using a **GetBaseReportRequest**	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The Charging Station responds with a **TriggerMessageResponse**	**1.** The OCTT sends a **TriggerMessageRequest** With **requestedMessage** *SignV2GCertificate* **EVSE** *EVSE (having an seccId) returned in the GetReportResponse or omitted in case none is available*
	3 The Charging Station sends a **SignCertificateRequest**	**4.** The OCTT responds with a **SignCertificateResponse** With **status** *Accepted*
	6. The Charging Station responds with a **CertificateSignedResponse**	**5.** The OCTT sends a **CertificateSignedRequest** With **certificateChain** *<Certificate generated from the received CSR from step 3 and signed by SubCA2 or SubCA (if SubCA2 does not exist) certificate from the provided V2G certificate chain>* **certificateType** *V2GCertificate*
	Note(s): *Steps 1, 2, 3, 4, 5, and 6 are repeated for all returned seccIds*	
Tool validations	* Step 2: Message: **TriggerMessageResponse** - **status** must be *Accepted* * Step 3: Message: **SignCertificateRequest** - **csr** must contain *<An CSR that meets the following requirements: The key must be at least 256 bits long. The received CSR must be transmitted as described in RFC 2986 and then encoded in Privacy-Enhanced Mail (PEM) format.>* The certificate can only be an ECDSA certificate (ISO15118 cannot be used with RSA). If an seccId is found the csr should contain the seccId in the CN. * Step 6: Message: **CertificateSignedResponse** - **status** must be *Accepted*	
	Post scenario validations: N/a	

3. Test Cases Charging Station Management System

3.1. General pre/post conditions & tool validations

General conditions/validations are overruled by testcase specific conditions/validations, unless specifically stated otherwise.

General pre conditions:

The following pre conditions apply to all test cases, unless explicitly mentioned otherwise.

- The Configuration variable **TxCtrlr.TxStartPoint** is *"EVConnected,Authorized"*
- The Configuration variable **TxCtrlr.TxStopPoint** is *"EVConnected"*
- The Configuration variable **AuthCtrlr.AuthEnabled** is *true*
- The Configuration variable **AuthCtrlr.AuthorizeRemoteStart** is *false*
- The Configuration variable **AdditionalRootCertificateCheck** is *false*
- The Configuration variable **AllowNewSessionsPendingFirmwareUpdate** is *false*
- The Configuration variable **AlignedDataSendDuringIdle** is *false*

General tool rules/validations:

- The list of ChargingSchedulePeriod elements in a chargingSchedule SHALL be ordered by increasing values of ChargingSchedulePeriod.startPeriod. This means the list is in chronological order.
- The CSMS SHALL NOT set phaseToUse in a SetChargingProfileRequest when numberPhases is other than 1.

3.2. A Security

Table 448. Test Case Id: TC_A_01_CSMS

Test case name	**Basic Authentication - Valid username/password combination**	
Test case Id	TC_A_01_CSMS	
Use case Id(s)	A00, B01	
Requirement(s)	A00.FR.204, B01.FR.02	
System under test	CSMS	
Description	The Charging Station uses Basic authentication to authenticate itself to the CSMS, when using security profile 1 or 2.	
Purpose	To verify whether the CSMS is able to validate the (valid) Basic authentication credentials provided by the Charging Station at the connection request.	
Prerequisite(s)	The CSMS supports security profile 1 and/or 2	
Before (Preparations)	**Configuration State:** The CSMS must have a password configured that equals the configured BasicAuthPassword at the OCTT.	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The OCTT sends a HTTP upgrade request with an Authorization header, containing a username/password combination. Note(s): *- The Authorization header is formatted as follows: AUTHORIZATION: Basic <Base64 encoded(<Configured ChargingStationId>:<Configured BasicAuthPassword>)>*	**2.** The CSMS validates the username/password combination AND upgrades the connection to a (secured) WebSocket connection.
	3. The OCTT sends a **BootNotificationRequest**	**4.** The CSMS responds with a **BootNotificationResponse**
	5. The OCTT notifies the CSMS about the current state of all connectors.	**6.** The CSMS responds accordingly.
Tool validations	* Step 4: Message: **BootNotificationResponse** - **status** must be *Accepted*	
	Post scenario validations: N/a	

Table 449. Test Case Id: TC_A_02_CSMS

Test case name	**Basic Authentication - Username does not equal ChargingStationId**	
Test case Id	TC_A_02_CSMS	
Use case Id(s)	A00	
Requirement(s)	A00.FR.204	
System under test	CSMS	
Description	The Charging Station uses Basic authentication to authenticate itself to the CSMS, when using security profile 1 or 2.	
Purpose	To verify whether the CSMS is able to validate the (invalid) Basic authentication credentials provided by the Charging Station at the connection request.	
Prerequisite(s)	The CSMS supports security profile 1 and/or 2	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The OCTT sends a HTTP upgrade request with an Authorization header, containing a username/password combination. Note(s): *- The Authorization header is formatted as follows: AUTHORIZATION: Basic <Base64 encoded(<Configured ChargingStationId> + Invalid:<Configured basicAuthPassword>)>*	**2.** The CSMS validates the username/password combination AND rejects the connection upgrade request.
Tool validations	N/a	
	Post scenario validations: N/a	

Table 450. Test Case Id: TC_A_03_CSMS

Test case name	Basic Authentication - Invalid password	
Test case Id	TC_A_03_CSMS	
Use case Id(s)	A00	
Requirement(s)	A00.FR.204	
System under test	CSMS	
Description	The Charging Station uses Basic authentication to authenticate itself to the CSMS, when using security profile 1 or 2.	
Purpose	To verify whether the CSMS is able to validate the (invalid) Basic authentication credentials provided by the Charging Station at the connection request.	
Prerequisite(s)	The CSMS supports security profile 1 and/or 2	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The OCTT sends a HTTP upgrade request with an Authorization header, containing a username/password combination. Note(s): *- The Authorization header is formatted as follows: AUTHORIZATION: Basic <Base64 encoded(<Configured ChargingStationId>:<randomly chosen identifierString with a sufficiently high entropy, consisting of minimum 16 and maximum 40 characters (alpha-numeric characters and the special characters allowed by identifierString)>)>*	**2.** The CSMS validates the username/password combination AND rejects the connection upgrade request.
Tool validations	N/a	
	Post scenario validations: N/a	

Table 451. Test Case Id: TC_A_04_CSMS

Test case name	**TLS - server-side certificate - Valid certificate**
Test case Id	TC_A_04_CSMS
Use case Id(s)	A00
Requirement(s)	A00.FR.306,A00.FR.307,A00.FR.312,A00.FR.318,A00.FR.321,A00.FR.502,A00.FR.503,A00.FR.507,A00.FR.508,A00.FR.510
System under test	CSMS
Description	The CSMS uses a server-side certificate to identify itself to the Charging Station, when using security profile 2 or 3.
Purpose	To verify whether the CSMS is able to provide a valid server certificate and setup a secured WebSocket connection.
Prerequisite(s)	The CSMS supports security profile 2 and/or 3
Before (Preparations)	**Configuration State:** N/a
	Memory State: N/a
	Reusable State(s): N/a

Test case name	**TLS - server-side certificate - Valid certificate**	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The OCTT terminates the connection and initiates a TLS handshake and sends a Client Hello to the CSMS.	**2.** The CSMS responds with a Server Hello With the *<Configured server certificate>*
	3. The OCTT performs the following actions: Send client certificate Client Key Exchange Certificate verify Change Cipher Spec Finished Note(s): *- The client certificate is only sent when the CSMS uses security profile 3.*	**4.** The CSMS performs the following actions: Change Cipher Spec Finished
	5. The OCTT sends a HTTP upgrade request to the CSMS Note(s): *- The HTTP request only contains a username/password combination when the CSMS uses security profile 2.*	**6.** The CSMS upgrades the connection to a (secured) WebSocket connection.
	7. The OCTT sends a **BootNotificationRequest** with **reason** *PowerUp* **chargingStation.model** *<Configured model>* **chargingStation.vendorName** *<Configured vendorName>*	**8.** The CSMS responds with a **BootNotificationResponse**
	9. The OCTT notifies the CSMS about the current state of all connectors. Message: **StatusNotificationRequest** - **connectorStatus** *Available* Message: **NotifyEventRequest** - **trigger** *Delta* - **actualValue** *"Available"* - **component.name** *"Connector"* - **variable.name** *"AvailabilityState"*	**10.** The CSMS responds accordingly.

Test case name	**TLS - server-side certificate - Valid certificate**
Tool validations	* Step 3: The OCTT validates the following before finishing the TLS handshake: - The CSMS must use TLS version 1.2 or above At least the following set of cipher suites must be supported: TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 AND TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 AND TLS_RSA_WITH_AES_128_GCM_SHA256 AND TLS_RSA_WITH_AES_256_GCM_SHA384 - When using RSA or DSA the key must be at least 2048 bits long. and when using elliptic curve cryptography the key must be at least 224 bits long. - The received server side certificate must be transmitted in the X.509 format encoded in Privacy-Enhanced Mail (PEM) format. - The certificate must include a serial number. - The subject field of the certificate must contain a commonName RDN which consists of the FQDN of the endpoint of the server. *NOTE: If one of the above validations fails, the OCTT can still proceed with the next steps of the testcase (if it is able to), but the testcase will FAIL and the OCTT reports why it failed.* * Step 8: Message: **BootNotificationResponse** with **status** *Accepted*
	Post scenario validations: N/a

Table 452. Test Case Id: TC_A_06_CSMS

Test case name	**TLS - server-side certificate - TLS version too low**	
Test case Id	TC_A_06_CSMS	
Use case Id(s)	A00	
Requirement(s)	A00.FR.314,A00.FR.315,A00.FR.409,A00.FR.416,A00.FR.417,A00.FR.418	
System under test	CSMS	
Description	The CSMS uses a server-side certificate to identify itself to the Charging Station, when using security profile 2 or 3.	
Purpose	To verify whether the CSMS is able to terminate the connection when it notices the used TLS version is lower than 1.2.	
Prerequisite(s)	The CSMS supports security profile 2 and/or 3	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The OCTT terminates the connection and initiates a TLS handshake with a TLS version lower than 1.2 and sends a Client Hello to the CSMS.	**2.** The CSMS notices that the TLS version is lower than 1.2 and terminates the connection.
	3. The OCTT initiates a TLS handshake with TLS version 1.2 or higher and sends a Client Hello to the CSMS.	**4.** The CSMS responds with a Server Hello With the *<Configured server certificate>*
	5. The OCTT performs the following actions: Send client certificate Client Key Exchange Certificate verify Change Cipher Spec Finished Note(s): *- The client certificate is only sent when the CSMS uses security profile 3.*	**6.** The CSMS performs the following actions: Change Cipher Spec Finished
	7. The OCTT sends a HTTP upgrade request to the CSMS Note(s): *- The HTTP request only contains a username/password combination when the CSMS uses security profile 2.*	**8.** The CSMS upgrades the connection to a (secured) WebSocket connection.
	9. The OCTT sends a **BootNotificationRequest** with **reason** *PowerUp* **chargingStation.model** *<Configured model>* **chargingStation.vendorName** *<Configured vendorName>*	**10.** The CSMS responds with a **BootNotificationResponse**

Test case name	**TLS - server-side certificate - TLS version too low**	
	11. The OCTT notifies the CSMS about the current state of all connectors. Message: **StatusNotificationRequest** - **connectorStatus** *Available* Message: **NotifyEventRequest** - **trigger** *Delta* - **actualValue** *"Available"* - **component.name** *"Connector"* - **variable.name** *"AvailabilityState"*	**12.** The CSMS responds accordingly.
Tool validations	* Step 10: Message: **BootNotificationResponse** - **status** *Accepted*	
	Post scenario validations: N/a	

Table 453. Test Case Id: TC_A_07_CSMS

Test case name	**TLS - Client-side certificate - valid certificate**	
Test case Id	TC_A_07_CSMS	
Use case Id(s)	A00	
Requirement(s)	A00.FR.409,A00.FR.410,A00.FR.415,A00.FR.416,A00.FR.421	
System under test	CSMS	
Description	The Charging Station uses a client-side certificate to identify itself to the CSMS, when using security profile 3.	
Purpose	To verify whether the CSMS is able to receive a client certificate provided by a Charging Station and setup a secured WebSocket connection.	
Prerequisite(s)	The CSMS supports security profile 3	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The OCTT terminates the connection and initiates a TLS handshake and sends a Client Hello to the CSMS.	**2.** The CSMS responds with a Server Hello With the *<Configured server certificate>*
	3. The OCTT performs the following actions: Send *<Configured client certificate>* Client Key Exchange Certificate verify Change Cipher Spec Finished	**4.** The CSMS performs the following actions: Change Cipher Spec Finished
	5. The OCTT sends a HTTP upgrade request to the CSMS	**6.** The CSMS upgrades the connection to a (secured) WebSocket connection.
	7. The OCTT sends a **BootNotificationRequest** with **reason** *PowerUp* **chargingStation.model** *<Configured model>* **chargingStation.vendorName** *<Configured vendorName>*	**8.** The CSMS responds with a **BootNotificationResponse**
	9. The OCTT notifies the CSMS about the current state of all connectors. Message: **StatusNotificationRequest** - **connectorStatus** *Available* Message: **NotifyEventRequest** - **trigger** *Delta* - **actualValue** *"Available"* - **component.name** *"Connector"* - **variable.name** *"AvailabilityState"*	**10.** The CSMS responds accordingly.

Test case name	**TLS - Client-side certificate - valid certificate**
Tool validations	* Step 3: The OCTT validates the following before finishing the TLS handshake: - The CSMS must use TLS version 1.2 or above At least the following set of cipher suites must be supported: TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 AND TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 AND TLS_RSA_WITH_AES_128_GCM_SHA256 AND TLS_RSA_WITH_AES_256_GCM_SHA384 * Step 8: Message: **BootNotificationResponse** with **status** *Accepted*
	Post scenario validations: N/a

Table 454. Test Case Id: TC_A_08_CSMS

Test case name	**TLS - Client-side certificate - Invalid certificate**	
Test case Id	TC_A_08_CSMS	
Use case Id(s)	A00	
Requirement(s)	A00.FR.405,A00.FR.407,A00.FR.409,A00.FR.410	
System under test	CSMS	
Description	The Charging Station uses a client-side certificate to identify itself to the CSMS, when using security profile 3.	
Purpose	To verify whether the CSMS is able to terminate the connection when the received client certificate is invalid.	
Prerequisite(s)	- The CSMS supports security profile 3 - This testcase can be executed multiple times, using different kinds of invalid certificates: Unknown certificate expired certificate certificate with commonName that does not equal the serial number of the Charging Station.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The OCTT initiates a TLS handshake and sends a Client Hello to the CSMS.	**2.** The CSMS responds with a Server Hello With a server certificate
	3. The OCTT performs the following actions: Send *<Configured invalid client certificate>* Client Key Exchange Certificate verify Change Cipher Spec Finished	**4.** The CSMS deems the client certificate invalid and terminates the connection.
	5. The OCTT initiates a TLS handshake and sends a Client Hello to the CSMS.	**6.** The CSMS responds with a Server Hello With a server certificate
	7. The OCTT performs the following actions: Send *<Configured client certificate>* Client Key Exchange Certificate verify Change Cipher Spec Finished	**8.** The CSMS performs the following actions: Change Cipher Spec Finished
	9. The OCTT sends a HTTP upgrade request to the CSMS	**10.** The CSMS upgrades the connection to a (secured) WebSocket connection.
	11. The OCTT sends a **BootNotificationRequest** with **reason** *PowerUp* **chargingStation.model** *<Configured model>* **chargingStation.vendorName** *<Configured vendorName>*	**12.** The CSMS responds with a **BootNotificationResponse**

Test case name	**TLS - Client-side certificate - Invalid certificate**	
	13. The OCTT notifies the CSMS about the current state of all connectors. Message: **StatusNotificationRequest** - **connectorStatus** *Available* Message: **NotifyEventRequest** - **trigger** *Delta* - **actualValue** *"Available"* - **component.name** *"Connector"* - **variable.name** *"AvailabilityState"*	**14.** The CSMS responds accordingly.
Tool validations	* Step 12: Message: **BootNotificationResponse** with **status** *Accepted*	
	Post scenario validations: N/a	

Table 455. Test Case Id: TC_A_09_CSMS

Test case name	**Update Charging Station Password for HTTP Basic Authentication - Accepted**	
Test case Id	TC_A_09_CSMS	
Use case Id(s)	A01	
Requirement(s)	A01.FR.02, A01.FR.03	
System under test	CSMS	
Description	This test case defines how to use the BasicAuthPassword, the password used to authenticate Charging Stations in security profile 1 (Basic Authentication) and security profile 2 (TLS with Basic Authentication)	
Purpose	To verify if the CSMS is able to successfully set the new BasicAuthPassword and only accepts the new credentials as described at the OCPP specification.	
Prerequisite(s)	The CSMS supports security profile 1 and/or 2	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **SetVariablesResponse** with **status** *Accepted*	**1.** The CSMS sends a **SetVariablesRequest** with: **setVariableData**[1]: - **variable.name** = *"BasicAuthPassword"* - **component.name** = *"SecurityCtrlr"* - **attributeValue** = *"<NewPassword>"*
	3. The OCTT sends a HTTP upgrade request with an Authorization header, containing a username/password combination (with the new *BasicAuthPassword*). Note(s): *- The Authorization header is formatted as follows: AUTHORIZATION: Basic <Base64 encoded(<Configured ChargingStationId>:<NEW BasicAuthPassword>)>*	**4.** The CSMS validates the username/password combination AND upgrades the connection to a (secured) WebSocket connection.
	5. The OCTT sends a **BootNotificationRequest**	**6.** The CSMS responds with a **BootNotificationResponse**
	7. The OCTT notifies the CSMS about the current state of all connectors.	**8.** The CSMS responds accordingly.
Tool validations	* Step 1: Message: **SetVariableRequest** - **variable.name** = *"BasicAuthPassword"* - **component.name** = *"SecurityCtrlr"* * Step 6: Message: **BootNotificationResponse** - **status** must be *Accepted*	
	Post scenario validations: N/a	

Table 456. Test Case Id: TC_A_10_CSMS

Test case name	**Update Charging Station Password for HTTP Basic Authentication - Rejected**	
Test case Id	TC_A_10_CSMS	
Use case Id(s)	A01	
Requirement(s)	A01.FR.02, A01.FR.04, A01.FR.05	
System under test	CSMS	
Description	This test case defines how to use the BasicAuthPassword, the password used to authenticate Charging Stations in security profile 1 (Basic Authentication) and security profile 2 (TLS with Basic Authentication)	
Purpose	To verify if the CSMS keeps accepting the old credentials and keeps communication when the new BasicAuthPassword is rejected as described at the OCPP specification.	
Prerequisite(s)	The CSMS supports security profile 1 and/or 2	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **SetVariablesResponse** with **status** *Rejected*	**1.** The CSMS sends a **SetVariablesRequest** with: **setVariableData**[1]: - **variable.name** = *"BasicAuthPassword"* - **component.name** = *"SecurityCtrlr"* - **attributeValue** = *"<NewPassword>"*
	3. The OCTT sends a HTTP upgrade request with an Authorization header, containing a username/password combination (with the old *BasicAuthPassword*). Note(s): *- The Authorization header is formatted as follows: AUTHORIZATION: Basic <Base64 encoded(<Configured ChargingStationId>:<OLD Configured BasicAuthPassword>)>*	**4.** The CSMS validates the username/password combination AND upgrades the connection to a (secured) WebSocket connection.
	5. The OCTT sends a **BootNotificationRequest**	**6.** The CSMS responds with a **BootNotificationResponse**
	7. The OCTT notifies the CSMS about the current state of all connectors.	**8.** The CSMS responds accordingly.
Tool validations	* Step 1: Message: **SetVariableRequest** - **variable.name** = *"BasicAuthPassword"* - **component.name** = *"SecurityCtrlr"* * Step 6: Message: **BootNotificationResponse** - **status** must be *Accepted*	
	Post scenario validations: N/a	

Table 457. Test Case Id: TC_A_11_CSMS

Test case name	**Update Charging Station Certificate by request of CSMS - Success - Charging Station Certificate**	
Test case Id	TC_A_11_CSMS	
Use case Id(s)	A02 & F06	
Requirement(s)	A02.FR.11, A02.FR.14 & F06.FR.01	
System under test	CSMS	
Description	The CSMS is able to request the Charging Station to update its charging station certificate using the TriggerMessageRequest message.	
Purpose	To verify if the CSMS is able to request the Charging Station to update its Charging Station Certificate.	
Prerequisite(s)	The CSMS supports security profile 3	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *RenewChargingStationCertificate*	
Tool validations	N/a	
	Post scenario validations: N/a	

Table 458. Test Case Id: TC_A_12_CSMS

Test case name	**Update Charging Station Certificate by request of CSMS - Success - V2G Certificate**	
Test case Id	TC_A_12_CSMS	
Use case Id(s)	A02 & F06	
Requirement(s)	A02.FR.11 & F06.FR.01	
System under test	CSMS	
Description	The CSMS is able to request the Charging Station to update its charging station certificate using the TriggerMessageRequest message.	
Purpose	To verify if the CSMS is able to request the Charging Station to update its V2G Certificate.	
Prerequisite(s)	The CSMS supports ISO 15118.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **TriggerMessageResponse** With **status** *Accepted*	**1.** The CSMS sends a **TriggerMessageRequest**
	3 The OCTT sends a **SignCertificateRequest** With **csr** *Generated CSR based on:* - *<Configured Country>* - *<Configured Organization>* - *<Configured OrganizationalUnit>* **certificateType** *V2GCertificate*	**4.** The CSMS responds with a **SignCertificateResponse**
	6. The OCTT responds with a **CertificateSignedResponse** With **status** *Accepted*	**5.** The CSMS sends a **CertificateSignedRequest**
Tool validations	* Step 1: Message: **TriggerMessageRequest** - **requestedMessage** *SignV2GCertificate* * Step 4: Message: **SignCertificateResponse** - **status** *Accepted* * Step 5: Message: **CertificateSignedRequest** - **certificateChain** *<Certificate generated from the received CSR from step 3 and signed by the V2G Root or SubCA certificate from the configured V2G certificate chain>* *NOTE: The OCTT will validate the certificate, but if the following validation fail, the testcase will NOT FAIL, because generating the certificate is probably not be done by the CSMS.* - The key must be at least 224 bits long. - The received certificate must be transmitted in the X.509 format encoded in Privacy-Enhanced Mail (PEM) format.	
	Post scenario validations: N/a	

Table 459. Test Case Id: TC_A_13_CSMS

Test case name	**Update Charging Station Certificate by request of CSMS - Success - Combined Certificate**	
Test case Id	TC_A_13_CSMS	
Use case Id(s)	A02	
Requirement(s)	A02.FR.11	
System under test	CSMS	
Description	The CSMS is able to request the Charging Station to update its charging station certificate using the TriggerMessageRequest message.	
Purpose	To verify if the CSMS is able to request the Charging Station to update a its combined V2G / Charging Station Certificate.	
Prerequisite(s)	- The CSMS supports security profile 3 - The CSMS supports ISO 15118.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **TriggerMessageResponse** With **status** *Accepted*	**1.** The CSMS sends a **TriggerMessageRequest**
	3 The OCTT sends a **SignCertificateRequest** With **csr** *<Configured CSR>*	**4.** The CSMS responds with a **SignCertificateResponse**
	6. The OCTT responds with a **CertificateSignedResponse** With **status** *Accepted*	**5.** The CSMS sends a **CertificateSignedRequest**
Tool validations	* Step 1: Message: **TriggerMessageRequest** - **requestedMessage** *SignCombinedCertificate* * Step 4: Message: **SignCertificateResponse** - **status** *Accepted* * Step 5: Message: **CertificateSignedRequest** - **certificateChain** *<Certificate generated from the received CSR from step 3 and signed by the V2G Root or SubCA certificate from the configured V2G certificate chain>* *NOTE: The OCTT will validate the certificate, but if the following validation fail, the testcase will NOT FAIL, because generating the certificate is probably not be done by the CSMS.* - The key must be at least 224 bits long. - The received certificate must be transmitted in the X.509 format encoded in Privacy-Enhanced Mail (PEM) format. - The certificate must include a serial number. - The subject field of the certificate must contain a commonName RDN which consists of the unique serial number of the Charging Station.	
	Post scenario validations: N/a	

Table 460. Test Case Id: TC_A_14_CSMS

Test case name	**Update Charging Station Certificate by request of CSMS - Invalid certificate**	
Test case Id	TC_A_14_CSMS	
Use case Id(s)	A02	
Requirement(s)	N/a	
System under test	CSMS	
Description	The CSMS is able to request the Charging Station to update its charging station certificate using the TriggerMessageRequest message.	
Purpose	To verify if the CSMS is able to handle a Charging Station rejecting the new Charging Station certificate.	
Prerequisite(s)	The CSMS supports security profile 3	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **TriggerMessageResponse** With **status** *Accepted*	**1.** The CSMS sends a **TriggerMessageRequest**
	3 The OCTT sends a **SignCertificateRequest** With **csr** *<Configured CSR>* **certificateType** *ChargingStationCertificate*	**4.** The CSMS responds with a **SignCertificateResponse**
	6. The OCTT responds with a **CertificateSignedResponse** With **status** *Rejected*	**5.** The CSMS sends a **CertificateSignedRequest**
Tool validations	* Step 1: Message: **TriggerMessageRequest** - **requestedMessage** *SignChargingStationCertificate* * Step 4: Message: **SignCertificateResponse** - **status** *Accepted*	
	Post scenario validations: N/a	

Table 461. Test Case Id: TC_A_19_CSMS

Test case name	**Upgrade Charging Station Security Profile - Accepted**	
Test case Id	TC_A_19_CSMS	
Use case Id(s)	A05	
Requirement(s)	A05.FR.04, A05.FR.07	
System under test	CSMS	
Description	The CSMS updates the connection details on the Charging Station, to increase the security profile level.	
Purpose	To verify if the CSMS is able to set a new network connection profile at one of the by the Charging Station defined configuration slots with a higher security profile than currently configured.	
Prerequisite(s)	- Security profile must be set to 1 or 2. - If Security profile is set to 1, then a trusted certificate must be installed.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: If configured <Security profile> is 2, then *RenewChargingStationCertificate*	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Request the CSMS to set a new NetworkConnectionProfile with a security profile level one higher than currently configured*	
	2. The OCTT responds with a **SetNetworkProfileResponse** With **status** *Accepted*	**1.** The CSMS sends a **SetNetworkProfileRequest**
	Manual Action: *Request the CSMS to change the NetworkConfigurationPriority to one that contains the configurationSlot of the new NetworkConnectionProfile from step 1*	
	4. The OCTT responds with a **SetVariablesResponse** with **status** *Accepted*	**3.** The CSMS sends a **SetVariablesRequest**
	Manual Action: *Request the CSMS to reboot the Charging Station*	
	6. The OCTT responds with a **ResetResponse** with **status** *Accepted*	**5.** The CSMS sends a **ResetRequest**
	7. The OCTT reconnects to the CSMS with security profile is <Configured securityProfile + 1>	**8.** The CSMS accepts the connection attempt.
	9. Execute **Reusable State** *Booted*	
	10. The OCTT reconnects to the CSMS with security profile is <Configured securityProfile>	**11.** The CSMS shall not accept the connection attempt.
Tool validations	* Step 1: Message **SetNetworkProfileRequest** - **connectionData.messageTimeout** *<Configured messageTimeout>* - **connectionData.ocppCsmsUrl** *<Configured ocppCsmsUrl>* - **connectionData.ocppInterface** *<Configured ocppInterface>* - **connectionData.ocppTransport** *JSON* - **connectionData.ocppVersion** *OCPP20* - **connectionData.securityProfile** *<Configured securityProfile + 1>* * Step 3: Message **SetVariablesRequest** **setVariableData**: - **variable.name** = *"NetworkConfigurationPriority"* - **component.name** = *"OCPPCommCtrlr"* - **attributeValue** = *<contains configurationSlot provided at step 1>*	
	Post scenario validations: - N/a	

3.3. B Provisioning

Table 462. Test Case Id: TC_B_01_CSMS

Test case name	**Cold Boot Charging Station - Accepted**	
Test case Id	TC_B_01_CSMS	
Use case Id(s)	B01	
Requirement(s)	B01.FR.02	
System under test	CSMS	
Description	The booting mechanism allows a Charging Station to provide some general information about the Charging Station to the CSMS on startup AND it allows the Charging Station to request whether it is allowed to start sending other OCPP messages.	
Purpose	To verify whether the CSMS is able to accept the communications of a registered Charging Station.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *Booted*	
Tool validations	N/a	
	Post scenario validations: N/a	

Table 463. Test Case Id: TC_B_02_CSMS

Test case name	**Cold Boot Charging Station - Pending**	
Test case Id	TC_B_02_CSMS	
Use case Id(s)	B02	
Requirement(s)	B02.FR.01, B02.FR.06	
System under test	CSMS	
Description	The booting mechanism allows a Charging Station to provide some general information about the Charging Station to the CSMS on startup AND it allows the Charging Station to request whether it is allowed to start sending other OCPP messages. The CSMS may respond to the BootNotificationRequest with status *Pending*. The *Pending* status can indicate that the CSMS wants to retrieve or set certain information on the Charging Station before it will accept the Charging Station.	
Purpose	To verify whether the CSMS is able to accept the communications of a registered Charging Station.	
Prerequisite(s)	The CSMS is configured to first respond to a BootNotificationRequest with **status** *Pending*.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The OCTT sends a **BootNotificationRequest** with **reason** *PowerUp* **chargingStation.model** *<Configured model>* **chargingStation.vendorName** *<Configured vendorName>*	**2.** The CSMS responds with a **BootNotificationResponse**
	Note(s): - If the interval in the BootNotificationResponse equals 0, the OCTT will wait *<Configured heartbeatInterval>* seconds, before sending another BootNotificationRequest. - If the interval in the BootNotificationResponse > 0, the OCTT will wait <Interval provided at the BootNotificationResponse> seconds, before sending another BootNotificationRequest. - During this interval, the CSMS may send messages to retrieve information from the Charging Station (as described in use cases B06, B07, B08) or change its configuration by SetVariablesRequest (as described in use case B05). The OCTT will respond to these messages.	
	3. The OCTT sends a **BootNotificationRequest** with **reason** *PowerUp* **chargingStation.model** *<Configured model>* **chargingStation.vendorName** *<Configured vendorName>*	**4.** The CSMS responds with a **BootNotificationResponse**
	5. The OCTT notifies the CSMS about the current state of all connectors. Message: **StatusNotificationRequest** with **connectorStatus** *Available* Message: **NotifyEventRequest** with **trigger** *Delta* **actualValue** *"Available"* **component.name** *"Connector"* **variable.name** *"AvailabilityState"*	**6.** The CSMS responds accordingly.

Test case name	**Cold Boot Charging Station - Pending**
Tool validations	* Step 2: Message: **BootNotificationResponse** - **status** *Pending* * Step 3: Message: **BootNotificationResponse** - **status** *Accepted*
	Post scenario validations: N/a

Table 464. Test Case Id: TC_B_30_CSMS

Test case name	**Cold Boot Charging Station - Pending/Rejected - SecurityError**	
Test case Id	TC_B_30_CSMS	
Use case Id(s)	B02/B03	
Requirement(s)	B02.FR.09, B03.FR.07	
System under test	CSMS	
Description	The booting mechanism allows a Charging Station to provide some general information about the Charging Station to the CSMS on startup AND it allows the Charging Station to request whether it is allowed to start sending other OCPP messages. The CSMS may respond to the BootNotificationRequest with status *Pending* or *Rejected*. During this state, the Charging Station is not allowed to send RPC Framework: CALL message that is NOT a BootNotificationRequest or in case of status *Pending*, a message triggered by one of the following messages: TriggerMessageRequest, GetBaseReportRequest, GetReportRequest.	
Purpose	To verify whether the CSMS is able to handle unauthorized messages from the Charging Station by responding with a SecurityError.	
Prerequisite(s)	The CSMS is configured to first respond to a BootNotificationRequest with **status** *Pending* or *Rejected*.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The OCTT sends a **BootNotificationRequest** with **reason** *PowerUp* **chargingStation.model** *<Configured model>* **chargingStation.vendorName** *<Configured vendorName>*	**2.** The CSMS responds with a **BootNotificationResponse**
	3. The OCTT notifies the CSMS about the current state of all connectors. Message: **StatusNotificationRequest** with **connectorStatus** *Available* Message: **NotifyEventRequest** with **trigger** *Delta* **actualValue** *"Available"* **component.name** *"Connector"* **variable.name** *"AvailabilityState"*	**4.** The CSMS responds with RPC Framework: CALLERROR: SecurityError.
Tool validations	* Step 2: Message: **BootNotificationResponse** - **status** *Pending* OR *Rejected*	
	Post scenario validations: N/a	

Table 465. Test Case Id: TC_B_31_CSMS

Test case name	**Cold Boot Charging Station - Pending/Rejected - TriggerMessage**	
Test case Id	TC_B_31_CSMS	
Use case Id(s)	B02, F06	
Requirement(s)	N/a	
System under test	CSMS	
Description	The booting mechanism allows a Charging Station to provide some general information about the Charging Station to the CSMS on startup AND it allows the Charging Station to request whether it is allowed to start sending other OCPP messages.	
Purpose	To verify whether the CSMS is able to send a TriggerMessageRequest to trigger a BootNotificationRequest, before the interval expired.	
Prerequisite(s)	The CSMS is configured to first respond to a BootNotificationRequest with **status** *Pending* or *Rejected*.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The OCTT sends a **BootNotificationRequest** with **reason** *PowerUp* **chargingStation.model** *<Configured model>* **chargingStation.vendorName** *<Configured vendorName>*	**2.** The CSMS responds with a **BootNotificationResponse**
	4. The OCTT responds with a **TriggerMessageResponse** with **status** *Accepted*	**3.** The CSMS sends a **TriggerMessageRequest**
	5. The OCTT sends a **BootNotificationRequest** with **reason** *Triggered* **chargingStation.model** *<Configured model>* **chargingStation.vendorName** *<Configured vendorName>*	**6.** The CSMS responds with a **BootNotificationResponse**
	7. The OCTT notifies the CSMS about the current state of all connectors. Message: **StatusNotificationRequest** with **connectorStatus** *Available* Message: **NotifyEventRequest** with **trigger** *Delta* **actualValue** *"Available"* **component.name** *"Connector"* **variable.name** *"AvailabilityState"*	**8.** The CSMS responds accordingly.
Tool validations	* Step 2: Message: **BootNotificationResponse** - **status** *Pending* OR *Rejected* * Step 3: Message: **TriggerMessageRequest** - **requestedMessage** *BootNotification* * Step 6: Message: **BootNotificationResponse** - **status** *Accepted*	
	Post scenario validations: N/a	

Table 466. Test Case Id: TC_B_06_CSMS

Test case name	**Get Variables - single value**	
Test case Id	TC_B_06_CSMS	
Use case Id(s)	B06	
Requirement(s)	B06.FR.01, B06.FR.02, B06.FR.03, B06.FR.04, B06.FR.10, B06.FR.11	
System under test	CSMS	
Description	Get the value of two of the required variables of OCPPCommCtrlr	
Purpose	To test getting single value using GetVariablesRequest for one of the mandatory component/variable combinations that must exist in the DM implementation.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. OCTT responds with: **GetVariablesResponse**	**1.** *Manually request CSMS to get data for:* - OCPPCommCtrlr.OfflineThreshold
Tool validations	* Step 1: Message: **GetVariablesRequest** with (in arbitrary order) **getVariableData**[0]: - **attributeType** is at least absent or **attributeType** = *Actual*, but *Target*, *MinSet*, and *MaxSet* are also allowed - **variable.name** = *"OfflineThreshold"* - **component.name** = *"OCPPCommCtrlr"*	
	Post scenario validations: Manually validate that CSMS has correctly read the requested variables.	

Table 467. Test Case Id: TC_B_07_CSMS

Test case name	**Get Variables - multiple values**	
Test case Id	TC_B_07_CSMS	
Use case Id(s)	B06	
Requirement(s)	B06.FR.01, B06.FR.02, B06.FR.03	
System under test	CSMS	
Description	Get the value of two of the required variables of OCPPCommCtrlr	
Purpose	To test getting multiple values using GetVariablesRequest for one of the mandatory component/variable combinations that must exist in the DM implementation.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. OCTT responds with: **GetVariablesResponse**	**1.** *Manually request CSMS to get data for:* - OCPPCommCtrlr.OfflineThreshold - AuthCtrlr.AuthorizeRemoteStart
Tool validations	* Step 1: Message: **GetVariablesRequest** with (in arbitrary order) **getVariableData**[0]: - **attributeType** is at least absent or **attributeType** = *Actual*, but *Target*, *MinSet*, and *MaxSet* are also allowed - **variable.name** = *"OfflineThreshold"* - **component.name** = *"OCPPCommCtrlr"* **getVariableData**[1]: - **attributeType** is at least absent or **attributeType** = *Actual*, but *Target*, *MinSet*, and *MaxSet* are also allowed - **variable.name** = *"AuthorizeRemoteStart"* - **component.name** = *"AuthCtrlr"*	
	Post scenario validations: Manually validate that CSMS has correctly read the requested variables.	

Table 468. Test Case Id: TC_B_08_CSMS

Test case name	**Get Variables - limit to maximum number of values**	
Test case Id	TC_B_08_CSMS	
Use case Id(s)	B06	
Requirement(s)	B06.FR.05	
System under test	CSMS	
Description	Do not request more variables than supported by `MaxItemsPerMessageGetVariables`.	
Purpose	To test that CSMS does not request more variables than the Charging Station reported to support in the variable `MaxItemsPerMessageGetVariables`.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** Configure (using getVariablesRequest) Component.Variable.Instance DeviceDataCtrlr.ItemsPerMessage.GetVariables at value 4.	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. OCTT responds with **GetVariablesResponse** with with a list of 4 **GetVariableResultType** items and a **GetVariableResponse** with 1 **GetVariableResultType** item.	**1.** *Manually request CSMS for 5 variables:* - DeviceDataCtrlr.ItemsPerMessage[GetReport] - DeviceDataCtrlr.ItemsPerMessage[GetVariables] - DeviceDataCtrlr.BytesPerMessage[GetReport] - DeviceDataCtrlr.BytesPerMessage[GetVariables] - AuthCtrlr.AuthorizeRemoteStart
Tool validations	* Step 1: Message: **GetVariablesRequest** for 4 variables and a **GetVariablesRequest** for 1 variable (in arbitrary order): for **component.name** = *"DeviceDataCtrlr"* - **variable.name** = *"ItemsPerMessage"* with **variable.instance** = *"GetReport"* - **variable.name** = *"ItemsPerMessage"* with **variable.instance** = *"GetVariables"* - **variable.name** = *"BytesPerMessage"* with **variable.instance** = *"GetReport"* - **variable.name** = *"BytesPerMessage"* with **variable.instance** = *"GetVariables"* and for **component.name** = *"AuthCtrlr"* - **variable.name** = *"AuthorizeRemoteStart"*	
	Post scenario validations: OCTT validates that not more than `ItemsPerMessageGetVariables` elements are requested in one **GetVariablesRequest** message by CSMS.	

Table 469. Test Case Id: TC_B_09_CSMS

Test case name	**Set Variables - single value**	
Test case Id	TC_B_09_CSMS	
Use case Id(s)	B05	
Requirement(s)	B05.FR.01, B05.FR.02, B05.FR.03, B05.FR.10, B05.FR.12	
System under test	CSMS	
Description	Set the value of one of the required variables of OCPPCommCtrlr	
Purpose	To test setting a single value using SetVariablesRequest for one of the mandatory component/variable combinations that must exist in the DM implementation.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. OCTT responds with: **SetVariablesResponse**	**1.** *Manually request CSMS to set data for:* - OCPPCommCtrlr.OfflineThreshold
Tool validations	* Step 1: Message: **SetVariablesRequest** with (in arbitraty order): **setVariableData**[1]: - **variable.name** = *"OfflineThreshold"* - **component.name** = *"OCPPCommCtrlr"* - **attributeValue** = *"123"* - **attributeType** is absent or **attributeType** = *Actual*	
	Post scenario validations: Manually validate that CSMS has correctly set the requested variables.	

Table 470. Test Case Id: TC_B_10_CSMS

Test case name	**Set Variables - multiple values**	
Test case Id	TC_B_10_CSMS	
Use case Id(s)	B05	
Requirement(s)	B05.FR.01, B05.FR.02, B05.FR.03	
System under test	CSMS	
Description	Set the value of two of the required variables of OCPPCommCtrlr	
Purpose	To test setting multiple values using SetVariablesRequest for one of the mandatory component/variable combinations that must exist in the DM implementation.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. OCTT responds with: **SetVariablesResponse**	**1.** *Manually request CSMS to set data for:* - OCPPCommCtrlr.OfflineThreshold - AuthCtrlr.AuthorizeRemoteStart+
Tool validations	* Step 1: Message: **SetVariablesRequest** with (in arbitraty order): **setVariableData**[1]: - **variable.name** = *"OfflineThreshold"* - **component.name** = *"OCPPCommCtrlr"* - **attributeValue** = *"123"* - **attributeType** is absent or **attributeType** = *Actual* **setVariableData**[2]: - **variable.name** = *"AuthorizeRemoteStart"* - **component.name** = *"AuthCtrlr"* - **attributeValue** = *"false"* - **attributeType** is absent or **attributeType** = *Actual*	
	Post scenario validations: Manually validate that CSMS has correctly set the requested variables.	

Table 471. Test Case Id: TC_B_12_CSMS

Test case name	**Get Base Report - ConfigurationInventory**	
Test case Id	TC_B_12_CSMS	
Use case Id(s)	B07	
Requirement(s)	B07.FR.07	
System under test	CSMS	
Description	CSMS requests a ConfigurationInventory base report.	
Purpose	To test that CSMS supports the ConfigurationInventory base report.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. OCTT responds with: **GetBaseReportResponse**	**1.** *Manually instruct CSMS to retrieve a ConfigurationInventory report.*
Tool validations	* Step 1: Message: **GetBaseReportRequest** with: - **requestId** has integer value >= 0 - **reportBase** = *ConfigurationInventory*	
	Post scenario validations: CSMS receives all **NotifyReportRequest** message for this *requestId* and is able to show the result of configuration inventory to an operator.	

Table 472. Test Case Id: TC_B_13_CSMS

Test case name	**Get Base Report - FullInventory**	
Test case Id	TC_B_13_CSMS	
Use case Id(s)	B07	
Requirement(s)	B07.FR.08	
System under test	CSMS	
Description	CSMS requests a FullInventory base report.	
Purpose	To test that CSMS supports the FullInventory base report.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. OCTT responds with: **GetBaseReportResponse**	**1.** *Manually instruct CSMS to retrieve a FullInventory report.*
Tool validations	* Step 1: **GetBaseReportRequest** with: - **requestId** has integer value >= 0 - **reportBase** = *FullInventory*	
	Post scenario validations: CSMS receives all **NotifyReportRequest** message for this *requestId* and is able to show the result of full inventory to an operator.	

Table 473. Test Case Id: TC_B_14_CSMS

Test case name	**Get Base Report - SummaryInventory**	
Test case Id	TC_B_14_CSMS	
Use case Id(s)	B07	
Requirement(s)	B07.FR.09	
System under test	CSMS	
Description	CSMS requests a SummaryInventory base report.	
Purpose	To test that CSMS supports the SummaryInventory base report.	
Prerequisite(s)	CSMS implementation supports the optional SummaryInventory report	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. OCTT responds with: **GetBaseReportResponse**	**1.** *Manually instruct CSMS to retrieve a SummaryInventory report.*
Tool validations	* Step 1: **GetBaseReportRequest** with: - **requestId** has integer value >= 0 - **reportBase** = *SummaryInventory*	
	Post scenario validations: CSMS receives all **NotifyReportRequest** message for this *requestId* and is able to show the result of summary inventory to an operator.	

Table 474. Test Case Id: TC_B_18_CSMS

Test case name	**Get Custom Report - with componentCriteria and component/variables**	
Test case Id	TC_B_18_CSMS	
Use case Id(s)	B08	
Requirement(s)	B08.FR.01, B08.FR.03	
System under test	CSMS	
Description	CSMS requests a report of components that match both the component criteria and the given list of components and variables.	
Purpose	To test that CSMS supports requesting a report for both the component criteria and a given list of components and optionally with variables and that it handles an empty result set.	
Prerequisite(s)		
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. OCTT responds with: **GetReportResponse** with **status** *EmptyResultSet*	**1.** *Manually instruct CSMS to get the value of:* - EVSE #1::AvailabiltyState - from all *Problem* components
	4. OCTT responds with: **GetReportResponse** with **status** *Accepted*	**3.** *Manually instruct CSMS to get the value of:* - EVSE #1::AvailabiltyState - from all *Available* components
	5. OCTT responds with: **NotifyReportRequest**	**6.** CSMS sends **NotifyReportResponse**
Tool validations	* Step 1: Message: **GetReportRequest** - **componentCriteria** = *Problem* - **componentVariable[0].component.name** = *"EVSE"* - **componentVariable[0].component.evse.id** = *1* - **componentVariable[0].variable.name** = *"AvailabilityState"*	
	* Step 3: Message: **GetReportRequest** - **componentCriteria** is *Available* - **componentVariable[0].component.name** = *"EVSE"* - **componentVariable[0].component.evse.id** = *1* - **componentVariable[0].variable.name** = *"AvailabilityState"*	
	Post scenario validations: N/A	

Table 475. Test Case Id: TC_B_20_CSMS

Test case name	**Reset Charging Station - Without ongoing transaction - OnIdle**	
Test case Id	TC_B_20_CSMS	
Use case Id(s)	B11	
Requirement(s)	B11.FR.04	
System under test	CSMS	
Description	This test case covers how the CSMS can request the Charging Station to reset itself by sending a ResetRequest without any ongoing transaction. This could for example be necessary if the Charging Station is not functioning correctly.	
Purpose	To verify if the CSMS is able to perform the reset mechanism as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Request the CSMS to reboot the Charging Station with **type** _OnIdle*	
	2. The OCTT responds with a **ResetResponse** with **status** *Accepted*	**1.** The CSMS sends a **ResetRequest**
	3. The OCTT sends a **BootNotificationRequest**	**4.** The CSMS responds with a **BootNotificationResponse**
	5. The OCTT notifies the CSMS about the current state of all connectors. Message: **StatusNotificationRequest** - **connectorStatus** *Available* Message: **NotifyEventRequest** - **trigger** *Delta* - **actualValue** *"Available"* - **component.name** *"Connector"* - **variable.name** *"AvailabilityState"*	**6.** The CSMS responds accordingly.
Tool validations	* Step 4: Message **BootNotificationResponse** - **status** *Accepted*	
	Post scenario validations: - N/a	

Table 476. Test Case Id: TC_B_21_CSMS

Test case name	**Reset Charging Station - With Ongoing Transaction - OnIdle**	
Test case Id	TC_B_21_CSMS	
Use case Id(s)	B12	
Requirement(s)	B12.FR.01, B12.FR.03, E07.FR.03	
System under test	CSMS	
Description	This test case covers how the CSMS can remotely request the Charging Station to reset itself by sending a ResetRequest during a transaction. When ResetRequest "OnIdle" is send the charging stations schedules a reboot after all transactions are stopped. This could for example be necessary if the Charging Station is not functioning correctly.	
Purpose	To verify if the CSMS is able to perform the reset mechanism as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): **State** is *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Request the CSMS to reboot the Charging Station with status OnIdle*	
	2. The OCTT responds with a **ResetResponse** with **status** *Scheduled*	**1.** The CSMS sends a **ResetRequest** with **status** *OnIdle*
	3. The OCTT sends a **TransactionEventRequest**. - **eventType** *Updated* - **triggerReason** *StopAuthorized* - **transactionInfo.chargingState** *EVConnected* - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>*	**4.** The CSMS responds with a **TransactionEventResponse**.
	5. The OCTT sends a **TransactionEventRequest**. - **eventType** *Ended* - **triggerReason** *EVCommunicationLost* - **transactionInfo.chargingState** *Idle* - **transactionInfo.stoppedReason** *EVDisconnected*	**6.** The CSMS responds with a **TransactionEventResponse**.
	7. The OCTT sends a **BootNotificationRequest** with **reason** *ScheduledReset*	**8.** The CSMS responds with a **BootNotificationResponse**
	9. The OCTT notifies the CSMS about the current state of all connectors. Message: **StatusNotificationRequest** - **connectorStatus** *Available* Message: **NotifyEventRequest** - **trigger** *Delta* - **actualValue** *"Available"* - **component.name** *"Connector"* - **variable.name** *"AvailabilityState"*	**10.** The CSMS responds accordingly.

Test case name	**Reset Charging Station - With Ongoing Transaction - OnIdle**
Tool validations	* Step 1: Message **ResetRequest** - **type** *OnIdle* * Step 8: Message **BootNotificationResponse** - **status** *Accepted*
	Post scenario validations: - N/a

Table 477. Test Case Id: TC_B_22_CSMS

Test case name	**Reset Charging Station - With Ongoing Transaction - Immediate**	
Test case Id	TC_B_22_CSMS	
Use case Id(s)	B12	
Requirement(s)	N/a	
System under test	CSMS	
Description	This test case covers how the CSMS can remotely request the Charging Station to reset itself by sending a ResetRequest during a transaction. When ResetRequest "Immediate" is send the charging stations will try to stop all transactions before rebooting. This could for example be necessary if the Charging Station is not functioning correctly.	
Purpose	To verify if the CSMS is able to perform the reset mechanism as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): **State** is *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Request the CSMS to reboot the Charging Station with status Immediate*	
	2. The OCTT responds with a **ResetResponse** with **status** *Accepted*	**1.** The CSMS sends a **ResetRequest** with **status** *Immediate*
	3. The OCTT sends a **TransactionEventRequest**. - **eventType** *Ended* - **triggerReason** *ResetCommand* - **transactionInfo.chargingState** *EVConnected* - **transactionInfo.stoppedReason** *ImmediateReset* - **idToken** is omitted	**4.** The CSMS responds with a **TransactionEventResponse**.
	5. The OCTT sends a **BootNotificationRequest** with **reason** *RemoteReset*	**6.** The CSMS responds with a **BootNotificationResponse**
	7. The OCTT notifies the CSMS about the current state of all connectors. For *<Configured connectorId>:* Message: **StatusNotificationRequest** - **connectorStatus** *Occupied* Message: **NotifyEventRequest** - **trigger** *Delta* - **actualValue** *"Occupied"* - **component.name** *"Connector"* - **variable.name** *"AvailabilityState"* For *<Other connector(s)>:* Message: **StatusNotificationRequest** - **connectorStatus** *Available* Message: **NotifyEventRequest** - **trigger** *Delta* - **actualValue** *"Available"* - **component.name** *"Connector"* - **variable.name** *"AvailabilityState"*	**8.** The CSMS responds accordingly.

Test case name	**Reset Charging Station - With Ongoing Transaction - Immediate**
Tool validations	* Step 1: Message **ResetRequest** - **type** *Immediate* * Step 6: Message **BootNotificationResponse** - **status** *Accepted*
	Post scenario validations: - N/a

Table 478. Test Case Id: TC_B_25_CSMS

Test case name	**Reset EVSE - Without ongoing transaction**	
Test case Id	TC_B_25_CSMS	
Use case Id(s)	B11	
Requirement(s)	B11.FR.04	
System under test	CSMS	
Description	This test case covers how the CSMS can request the Charging Station to reset an EVSE by sending a ResetRequest without any ongoing transaction. This could for example be necessary if the Charging Station is not functioning correctly.	
Purpose	To verify if the CSMS is able to perform the reset mechanism as described at the OCPP specification.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	<u>Manual Action</u>: *Request the CSMS to reboot an EVSE with status OnIdle*	
	2. The OCTT responds with a **ResetResponse** with **status** *Accepted*	**1.** The CSMS sends a **ResetRequest** with **status** *OnIdle* and **evseID** *<Configured evseId>*
Tool validations	* Step 1: Message **ResetRequest** - **type** *OnIdle* - **evseId** *<Configured evseId>*	
	Post scenario validations: - N/a	

Table 479. Test Case Id: TC_B_26_CSMS

Test case name	**Reset EVSE - With Ongoing Transaction - OnIdle**	
Test case Id	TC_B_26_CSMS	
Use case Id(s)	B12	
Requirement(s)	B12.FR.07	
System under test	CSMS	
Description	This test case covers how the CSMS can remotely request the Charging Station to reset an EVSE by sending a ResetRequest during a transaction. When ResetRequest "OnIdle" is send the charging stations schedules a reboot after all transactions are stopped. This could for example be necessary if the Charging Station is not functioning correctly.	
Purpose	To verify if the CSMS is able to perform the reset mechanism as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): **State** is *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	<u>Manual Action</u>: *Request the CSMS to reboot the charging EVSE with status OnIdle*	
	2. The OCTT responds with a **ResetResponse** with **status** *Scheduled*	**1.** The CSMS sends a **ResetRequest** with **status** *OnIdle* and **evseID** *<Configured evseId>*
	3. The OCTT sends a **TransactionEventRequest**. - **eventType** *Updated* - **triggerReason** *StopAuthorized* - **transactionInfo.chargingState** *EVConnected* - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>*	**4.** The CSMS responds with a **TransactionEventResponse**.
	5. The OCTT sends a **TransactionEventRequest**. - **eventType** *Ended* - **triggerReason** *EVCommunicationLost* - **transactionInfo.chargingState** *Idle* - **transactionInfo.stoppedReason** *EVDisconnected*	**6.** The CSMS responds with a **TransactionEventResponse**.
Tool validations	* Step 1: Message **ResetRequest** - **type** *OnIdle* - **evseId** *<Configured evseId>*	
	Post scenario validations: - N/a	

Table 480. Test Case Id: TC_B_27_CSMS

Test case name	**Reset EVSE - With Ongoing Transaction - Immediate**	
Test case Id	TC_B_27_CSMS	
Use case Id(s)	B12	
Requirement(s)	N/a	
System under test	CSMS	
Description	This test case covers how the CSMS can remotely request the Charging Station to reset an EVSE by sending a ResetRequest during a transaction. When ResetRequest "Immediate" is send the charging stations will try to stop all transactions before rebooting. This could for example be necessary if the Charging Station is not functioning correctly.	
Purpose	To verify if the CSMS is able to perform the reset mechanism as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): **State** is *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Request the CSMS to reboot the charging EVSE with status Immediate*	
	2. The OCTT responds with a **ResetResponse** with **status** *Accepted*	**1.** The CSMS sends a **ResetRequest** with **status** *Immediate* and **evseId** *<Configured evseId>*
	3. The OCTT sends a **TransactionEventRequest**. - **eventType** *Ended* - **triggerReason** *ResetCommand* - **transactionInfo.chargingState** *EVConnected* - **transactionInfo.stoppedReason** *ImmediateReset*	**4.** The CSMS responds with a **TransactionEventResponse**.
Tool validations	* Step 1: Message **ResetRequest** - **type** *Immediate* - **evseId** *<Configured evseId>*	
	Post scenario validations: N/a	

Table 481. Test Case Id: TC_B_42_CSMS

Test case name	**Set new NetworkConnectionProfile - Accepted**	
Test case Id	TC_B_42_CSMS	
Use case Id(s)	B09	
Requirement(s)	B09.FR.01	
System under test	CSMS	
Description	The CSMS updates the connection details on the Charging Station. For instance in preparation of a migration to a new CSMS.	
Purpose	To verify if the CSMS is able to set a new network connection profile at one of the by the Charging Station defined configuration slots.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **SetNetworkProfileResponse** With **status** *Accepted*	**1.** The CSMS sends a **SetNetworkProfileRequest**
Tool validations	* Step 1: Message **SetNetworkProfileRequest** - **configurationSlot** is *<Configured configurationSlot>* - **connectionData.messageTimeout** *<Configured messageTimeout>* - **connectionData.ocppCsmsUrl** *<Configured ocppCsmsUrl>* - **connectionData.ocppInterface** *<Configured ocppInterface>* - **connectionData.ocppTransport** *JSON* - **connectionData.ocppVersion** *OCPP20* - **connectionData.securityProfile** *<Configured securityProfile>*	
	Post scenario validations: - N/a	

Table 482. Test Case Id: TC_B_44_CSMS

Test case name	**Set new NetworkConnectionProfile - Failed**	
Test case Id	TC_B_44_CSMS	
Use case Id(s)	B09	
Requirement(s)	B09.FR.03	
System under test	CSMS	
Description	The CSMS updates the connection details on the Charging Station. For instance in preparation of a migration to a new CSMS.	
Purpose	To verify if the CSMS is able to handle a Charging Station responding with status Failed, when setting a new network connection profile at one of the by the Charging Station defined configuration slots.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **SetNetworkProfileResponse** With **status** *Failed*	**1.** The CSMS sends a **SetNetworkProfileRequest**
Tool validations	N/a	
	Post scenario validations: - N/a	

3.4. C Authorization

Table 483. Test Case Id: TC_C_02_CSMS

Test case name	**Local start transaction - Authorization Invalid/Unknown**	
Test case Id	TC_C_02_CSMS	
Use case Id(s)	C01, C04, C06	
Requirement(s)	C01.FR.07 OR C04.FR.01 OR C06.FR.04	
System under test	CSMS	
Description	When a Charging Station needs to charge an EV, it needs to authorize the EV Driver first at the CSMS before the charging can be started or stopped.	
Purpose	To verify whether the CSMS is able to report that an idToken is NOT valid.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The OCTT sends an **AuthorizeRequest** with **idToken.idToken** *<Configured invalid_idtoken_idtoken>* **idToken.type** *<Configured invalid_idtoken_type>*	**2.** The CSMS responds with an **AuthorizeResponse**
Tool validations	* Step 2: Message: **AuthorizeResponse** - **idTokenInfo.status** *Invalid* or *Unknown*	
	Post scenario validations: - N/a	

Table 484. Test Case Id: TC_C_06_CSMS

Test case name	**Local start transaction - Authorization Blocked**	
Test case Id	TC_C_06_CSMS	
Use case Id(s)	C01	
Requirement(s)	C01.FR.07	
System under test	CSMS	
Description	When a Charging Station needs to charge an EV, it needs to authorize the EV Driver first at the CSMS before the charging can be started or stopped.	
Purpose	To verify whether the CSMS is able to report that an idToken is Blocked.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** The IdToken configured as Blocked at the OCTT, must be set as Blocked at the CSMS.	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The OCTT sends an **AuthorizeRequest** with **idToken.idToken** *<Configured blocked_idtoken_idtoken>* **idToken.type** *<Configured blocked_idtoken_type>*	**2.** The CSMS responds with an **AuthorizeResponse**
Tool validations	* Step 2: Message: **AuthorizeResponse** - **idTokenInfo.status** *Blocked* or *Invalid*	
	Post scenario validations:	

Table 485. Test Case Id: TC_C_07_CSMS

Test case name	**Local start transaction - Authorization Expired**	
Test case Id	TC_C_07_CSMS	
Use case Id(s)	C01	
Requirement(s)	C01.FR.07	
System under test	CSMS	
Description	When a Charging Station needs to charge an EV, it needs to authorize the EV Driver first at the CSMS before the charging can be started or stopped.	
Purpose	To verify whether the CSMS is able to report that an idToken is Expired.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** The IdToken configured as Expired at the OCTT, must be set as Expired at the CSMS.	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The OCTT sends an **AuthorizeRequest** with **idToken.idToken** *<Configured expired_idtoken_idtoken>* **idToken.type** *<Configured expired_idtoken_type>*	**2.** The CSMS responds with an **AuthorizeResponse**
Tool validations	* Step 2: Message: **AuthorizeResponse** - **idTokenInfo.status** *Expired* or *Invalid*	
	Post scenario validations:	

Table 486. Test Case Id: TC_C_08_CSMS

Test case name	**Authorization through authorization cache - Accepted**	
Test case Id	TC_C_08_CSMS	
Use case Id(s)	C12	
Requirement(s)	C12_FR_03	
System under test	CSMS	
Description	This test case describes how the EV Driver is authorized to start a transaction while the Charging Station uses Cached IdToken. This enables the EV Driver to Online start a transaction by using the Authorization Cache in which the Charging Station can respond faster, as no AuthorizeRequest is being sent.	
Purpose	To verify if the CSMS is able to respond correctly when an idToken which has status "Accepted" in the charging stations cache is presented according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: State is *EVConnectedPreSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The OCTT sends a **TransactionEventRequest** with - **triggerReason** *Authorized* - **idToken** *<Valid id token configured in Authorization Cache>* - **eventType** *Updated* Note(s): - ***TxStartPoint*** *contains ParkingBayOccupancy*	**2.** The CSMS responds with a **TransactionEventResponse**
Tool validations	* Step 2: Message **TransactionEventResponse** - **idTokenInfo.status** *Accepted*	
	Post scenario validations: - N/a	

Table 487. Test Case Id: TC_C_20_CSMS

Test case name	**Authorization through authorization cache - Invalid**	
Test case Id	TC_C_20_CSMS	
Use case Id(s)	C12	
Requirement(s)	C12_FR_03	
System under test	CSMS	
Description	This test case describes how the EV Driver is authorized to start a transaction while the Charging Station uses Cached IdToken. This enables the EV Driver to Online start a transaction by using the Authorization Cache in which the Charging Station can respond faster, as no AuthorizeRequest is being sent.	
Purpose	To verify if the CSMS is able to respond correctly when an idToken, which has status "Invalid" in the charging stations cache but not in the CSMS, is presented according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: State is *EVConnectedPreSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The OCTT sends a **TransactionEventRequest** with - **triggerReason** *Authorized* - **idToken.idToken** *<Configured invalid_idtoken_idtoken>* - **idToken.type** *<Configured invalid_idtoken_type>* - **eventType** *Updated* Note(s): - ***TxStartPoint*** *contains ParkingBayOccupancy*	**2.** The CSMS responds with a **TransactionEventResponse**
Tool validations	* Step 2: Message **TransactionEventResponse** - **idTokenInfo.status** *Invalid* or *Unknown*	
	Post scenario validations: - N/a	

Table 488. Test Case Id: TC_C_37_CSMS

Test case name	**Clear Authorization Data in Authorization Cache - Accepted**	
Test case Id	TC_C_37_CSMS	
Use case Id(s)	C11	
Requirement(s)	N/a	
System under test	CSMS	
Description	This test case covers how the Charging Station autonomously stores a record of previously presented identifiers that have been successfully authorized by the CSMS in the Authorization Cache. (Successfully meaning: a response received on a message containing an IdToken)	
Purpose	To verify if the CSMS is able to request the Charging Station to clear all identifiers from the Authorization Cache according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	- N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **ClearCacheResponse** with **status** *Accepted*	**1.** The CSMS sends a **ClearCacheRequest**
Tool validations	- N/a	
	Post scenario validations: - N/a	

Table 489. Test Case Id: TC_C_38_CSMS

Test case name	**Clear Authorization Data in Authorization Cache - Rejected**	
Test case Id	TC_C_38_CSMS	
Use case Id(s)	C11	
Requirement(s)	N/a	
System under test	CSMS	
Description	This test case covers how the Charging Station autonomously stores a record of previously presented identifiers that have been successfully authorized by the CSMS in the Authorization Cache. (Successfully meaning: a response received on a message containing an IdToken)	
Purpose	To verify if the CSMS is able to request the Charging Station to clear all identifiers from the Authorization Cache according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	- N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **ClearCacheResponse** with **status** *Rejected*	**1.** The CSMS sends a **ClearCacheRequest**
Tool validations	- N/a	
	Post scenario validations: - N/a	

Table 490. Test Case Id: TC_C_39_CSMS

Test case name	**Authorization by GroupId - Success**	
Test case Id	TC_C_39_CSMS	
Use case Id(s)	C09	
Requirement(s)	C09_FR_02, C09_FR_03	
System under test	CSMS	
Description	This test case covers how a Charging Station can authorize an action for an EV Driver based on GroupId information. This could for example be used if 2 people regularly use the same EV: they can use their own IdToken (e.g. RFID card), and can deauthorize transactions that were started with the other idToken (with the same GroupId).	
Purpose	To verify if the CSMS is able to correctly handle the Authorization of idTokens with the same GroupId according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	- N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: Two valid idTokens with the same GroupId are configured	
	Reusable State(s): state is *EVConnectedPreSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The OCTT sends an **AuthorizeRequest** with **idToken.idToken** *<Configured valid_idtoken2_idtoken>* **idToken.type** *<Configured valid_idtoken2_type>*	**2.** The CSMS responds with an **AuthorizeResponse**
	3. The OCTT sends a **TransactionEventRequest** with - **triggerReason** *Authorized* - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>* if transaction was already started - **eventType** *Updated* else - **eventType** *Started*	**4.** The CSMS responds with a **TransactionEventResponse**
	5. Execute **Reusable State** *EnergyTransferStarted*	
	6. The OCTT sends an **AuthorizeRequest** with **idToken.idToken** *<Configured valid_idtoken2_idtoken>* **idToken.type** *<Configured valid_idtoken2_type>*	**7.** The CSMS responds with an **AuthorizeResponse**
	8. The OCTT sends a **TransactionEventRequest** with - **triggerReason** *StopAuthorized* - **idToken.idToken** *<Configured valid_idtoken2_idtoken>* - **idToken.type** *<Configured valid_idtoken2_type>* - **eventType** *Updated*	**9.** The CSMS responds with a **TransactionEventResponse**
	10. Execute **Reusable State** *EVConnectedPostSession*	
	11. Execute **Reusable State** *EVDisconnected*	

Test case name	**Authorization by GroupId - Success**
Tool validations	* Step 2: Message **AuthorizeResponse** - **idTokenInfo.status** *Accepted* - **idTokenInfo.groupIdToken.idToken** *<Configured groupIdToken>* * Step 4: Message **TransactionEventResponse** - **idTokenInfo.status** *Accepted* - **idTokenInfo.groupIdToken.idToken** *<Configured groupIdToken>* * Step 7: Message **AuthorizeResponse** - **idTokenInfo.status** *Accepted* - **idTokenInfo.groupIdToken.idToken** *<Configured groupIdToken>* * Step 9: Message **TransactionEventResponse** - **idTokenInfo.status** *Accepted* - **idTokenInfo.groupIdToken.idToken** *<Configured groupIdToken>*
	Post scenario validations: - N/a

Table 491. Test Case Id: TC_C_40_CSMS

Test case name	**Authorization by GroupId - Success with Local Authorization List**	
Test case Id	TC_C_40_CSMS	
Use case Id(s)	C09	
Requirement(s)	C09_FR_02, C09_FR_03	
System under test	CSMS	
Description	This test case covers how a Charging Station can authorize an action for an EV Driver based on GroupId information. This could for example be used if 2 people regularly use the same EV: they can use their own IdToken (e.g. RFID card), and can deauthorize transactions that were started with the other idToken (with the same GroupId).	
Purpose	To verify if the CSMS is able to correctly handle the Authorization of idTokens with the same GroupId which are located in the Local Authorization List according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	- N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: Two valid idTokens with same GroupId are configured	
	Reusable State(s): state is *EVConnectedPreSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The OCTT sends a **TransactionEventRequest** with - **triggerReason** *Authorized* - **idToken.idToken** *<Configured valid_idtoken_idtoken>* (with a configured GroupId) which is configured in the local Authorization List - **idToken.type** *<Configured valid_idtoken_type>* (with a configured GroupId) which is configured in the local Authorization List If transaction was already started - **eventType** *Updated* else - **eventType** *Started*	**2.** The CSMS responds with a **TransactionEventResponse**
	3. Execute **Reusable State** *EnergyTransferStarted*	
	5. The OCTT sends a **TransactionEventRequest** with - **triggerReason** *StopAuthorized* - **idToken.idToken** *<Configured valid_idtoken2_idtoken>* (with same configured GroupId) which is configured in the local Authorization List>_ - **idToken.type** *<Configured valid_idtoken2_type>* - **eventType** *Updated*	**6.** The CSMS responds with a **TransactionEventResponse**
	7. Execute **Reusable State** *EVConnectedPostSession*	
	8. Execute **Reusable State** *EVDisconnected*	
Tool validations	* Step 2: Message **TransactionEventResponse** - **idTokenInfo.status** *Accepted* - **idTokenInfo.groupIdToken.idToken** *<Configured groupIdToken>* * Step 6: Message **TransactionEventResponse** - **idTokenInfo.status** *Accepted* - **idTokenInfo.groupIdToken.idToken** *<Configured groupIdToken>*	
	Post scenario validations: - N/a	

Table 492. Test Case Id: TC_C_43_CSMS

Test case name	**Authorization by GroupId - Invalid status with Local Authorization List**	
Test case Id	TC_C_43_CSMS	
Use case Id(s)	C09	
Requirement(s)	C09_FR_02, C09_FR_03	
System under test	CSMS	
Description	This test case covers how a Charging Station can authorize an action for an EV Driver based on GroupId information. This could for example be used if 2 people regularly use the same EV: they can use their own IdToken (e.g. RFID card), and can deauthorize transactions that were started with the other idToken (with the same GroupId).	
Purpose	To verify if the CSMS is able to correctly handle the Authorization of idTokens with the same GroupId which are located in the Local Authorization List according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	- N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: Two known valid idTokens with same GroupId are configured.	
	Reusable State(s): state is *EVConnectedPreSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The OCTT sends a **TransactionEventRequest** with - **triggerReason** *Authorized* - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>* if transaction was already started - **eventType** *Updated* else - **eventType** *Started*	**2.** The CSMS responds with a **TransactionEventResponse**
	3. Execute **Reusable State** *EnergyTransferStarted*	
	4. The OCTT sends an **AuthorizeRequest** with - **idToken.idToken** *<Configured valid_idtoken2_idtoken>* - **idToken.type** *<Configured valid_idtoken2_type>*	**5.** The CSMS responds with an **AuthorizeResponse**
	6. The OCTT sends a **TransactionEventRequest** with - **triggerReason** *StopAuthorized* - **idToken.idToken** *<Configured valid_idtoken2_idtoken>* - **idToken.type** *<Configured valid_idtoken2_type>* - **eventType** *Updated*	**7.** The CSMS responds with a **TransactionEventResponse**
	8. Execute **Reusable State** *EVConnectedPostSession*	
	9. Execute **Reusable State** *EVDisconnected*	

Test case name	**Authorization by GroupId - Invalid status with Local Authorization List**
Tool validations	* Step 1: Message **TransactionEventResponse** - **idTokenInfo.status** *Accepted* - **idTokenInfo.groupIdToken.idToken** *<Configured groupIdToken>* * Step 5: Message **AuthorizeResponse** - **idTokenInfo.status** *Accepted* - **idTokenInfo.groupIdToken.idToken** *<Configured groupIdToken>* * Step 7: Message **TransactionEventResponse** - **idTokenInfo.status** *Accepted* - **idTokenInfo.groupIdToken.idToken** *<Configured groupIdToken>*
	Post scenario validations: - N/a

Table 493. Test Case Id: TC_C_47_CSMS

Test case name	**Stop Transaction with a Master Pass - With UI - All transactions**	
Test case Id	TC_C_47_CSMS	
Use case Id(s)	C16	
Requirement(s)	C16_FR_01	
System under test	CSMS	
Description	This test case covers how somebody with a Master Pass (User) can stop (selected) ongoing transactions, so the cable becomes unlocked. This Master Pass can be configured in: MasterPassGroupId. This could for example be usefull for Law Enforcement officials.	
Purpose	To verify if the CSMS is able to correctly respond on a request to stop all transactions when an idToken which has the MasterPass as GroupId is used and the user has selected to stop all transactions in the User Interface according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	- N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: An idToken with the MastersPass as GroupId is configured	
	Reusable State(s): State is *EnergyTransferStarted* for EVSE 1 with idToken valid idToken State is *EnergyTransferStarted* for EVSE 2 with idToken valid idToken2	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The OCTT sends an **AuthorizeRequest** with **idToken.idToken** *<Configured masterpass_idtoken_idtoken>* **idToken.type** *<Configured masterpass_idtoken_type>*	**2.** The CSMS responds with an **AuthorizeResponse**
	3. The OCTT sends a **TransactionEventRequest** with - **transactionInfo.stoppedReason** *MasterPass* - **idToken.idToken** *<Configured masterpass_idtoken_idtoken>* - **idToken.type** *<Configured masterpass_idtoken_type>* - **eventType** *Ended* for both EVSE	**4.** The CSMS responds with a **TransactionEventResponse** for both EVSE

Table 494. Test Case Id: TC_C_48_CSMS

Test case name	**Stop Transaction with a Master Pass - With UI - With UI - Specific transactions**	
Test case Id	TC_C_48_CSMS	
Use case Id(s)	C16	
Requirement(s)	C16_FR_01	
System under test	CSMS	
Description	This test case covers how somebody with a Master Pass (User) can stop (selected) ongoing transactions, so the cable becomes unlocked. This Master Pass can be configured in: MasterPassGroupId. This could for example be usefull for Law Enforcement officials.	
Purpose	To verify if the CSMS is able to correctly respond on a request to stop a transaction when an idToken which has the MasterPass as GroupId is used and the user has selected to stop one transaction in the User Interface according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	- N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: An idToken with the MastersPass as GroupId is configured	
	Reusable State(s): State is *EnergyTransferStarted* for all EVSE	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The OCTT sends an **AuthorizeRequest** with **idToken.idToken** *<Configured valid_idtoken_idtoken>* **idToken.type** *<Configured valid_idtoken_type>*	**2.** The CSMS responds with an **AuthorizeResponse**
	3. The OCTT sends a **TransactionEventRequest** with - **transactionInfo.stoppedReason** *MasterPass* - **idToken.idToken** *<Configured masterpass_idtoken_idtoken>* - **idToken.type** *<Configured masterpass_idtoken_type>* - **eventType** *Ended*	**4.** The CSMS responds with a **TransactionEventResponse**
Tool validations	* Step 2: Message **AuthorizeResponse** - **idTokenInfo.status** *Accepted* - **idTokenInfo.groupIdToken.idToken** *<Configured masterPassGroupId>* * Step 4: Message **TransactionEventResponse** - **idTokenInfo.status** *Accepted* - **idTokenInfo.groupIdToken.idToken** *<Configured masterPassGroupId>*	
	Post scenario validations: - N/a	

Table 495. Test Case Id: TC_C_49_CSMS

Test case name	**Stop Transaction with a Master Pass - Without UI**	
Test case Id	TC_C_49_CSMS	
Use case Id(s)	C16	
Requirement(s)	C16_FR_02	
System under test	CSMS	
Description	This test case covers how somebody with a Master Pass (User) can stop (selected) ongoing transactions, so the cable becomes unlocked. This Master Pass can be configured in: MasterPassGroupId. This could for example be usefull for Law Enforcement officials.	
Purpose	To verify if the CSMS is able to correctly respond on a request to stop all transactions when an idToken which has the MasterPass as GroupId is used and the Charging Station does not have a User Interface according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	- N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: An idToken with the MastersPass as GroupId is configured	
	Reusable State(s): State is *EnergyTransferStarted* for EVSE 1 with idToken valid idToken State is *EnergyTransferStarted* for EVSE 2 with idToken valid idToken2	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The OCTT sends an **AuthorizeRequest** with **idToken.idToken** *<Configured masterpass_idtoken_idtoken>* **idToken.type** *<Configured masterpass_idtoken_type>*	**2.** The CSMS responds with an **AuthorizeResponse**
	3. The OCTT sends a **TransactionEventRequest** with - **transactionInfo.stoppedReason** *MasterPass* - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>* - **eventType** *Ended* for both EVSE	**4.** The CSMS responds with a **TransactionEventResponse** for both EVSE

Table 496. Test Case Id: TC_C_50_CSMS

Test case name	**Authorization using Contract Certificates 15118 - Online - Local contract certificate validation - Accepted**	
Test case Id	TC_C_50_CSMS	
Use case Id(s)	C07	
Requirement(s)	C07.FR.04	
System under test	CSMS	
Description	The Charging Station is able to authorize with contract certificates when it supports ISO 15118.	
Purpose	To verify if the CSMS is able to validate the certificate hash data and the provided eMAID.	
Prerequisite(s)	- The configured eMAID is known by the CSMS as valid. - The contract certificate is valid. - iso15118CertificateHashData has a responder URL that points to an OCSP service for OCTT. - CSMS does not have a cached OCSP response for the contract certificate.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): State is *EVConnectedPreSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The OCTT sends an **AuthorizeRequest** With **idToken.idToken** *<Configured valid_idtoken_idtoken>* **idToken.type** *<Configured valid_idtoken_type>* **iso15118CertificateHashData** contains *<hashes from configured (V2G) certificate chain*	**2.** The CSMS sends an OCSP request to responder URL of **iso15118CertificateHashData** to check validity
	3. The OCTT OCSP service reponds that certificate is valid.	**4.** The CSMS responds with a **AuthorizeResponse**
	5. The OCTT sends a **TransactionEventRequest** With **triggerReason** *Authorized*	**6.** The CSMS responds with a **TransactionEventResponse**
	5. Execute **Reusable State** *EnergyTransferStarted*	
Tool validations	* Step 2: CSMS sends an OCSP request for **iso15118CertificateHashData** * Step 3: OCTT checks that received request for **iso15118CertificateHashData** is valid * Step 4: Message: **AuthorizeResponse** - **idTokenInfo.status** *Accepted* - **certificateStatus** *Accepted* * Step 4: Message: **TransactionEventResponse** - **idTokenInfo.status** *Accepted*	
	Post scenario validations: N/a	

Table 497. Test Case Id: TC_C_51_CSMS

Test case name	**Authorization using Contract Certificates 15118 - Online - Local contract certificate validation - Rejected**	
Test case Id	TC_C_51_CSMS	
Use case Id(s)	C07	
Requirement(s)	C07.FR.16	
System under test	CSMS	
Description	The Charging Station is able to authorize with contract certificates when it supports ISO 15118.	
Purpose	To verify if the CSMS is able to validate the certificate hash data and the provided eMAID.	
Prerequisite(s)	- The configured eMAID is known by the CSMS as valid. - The contract certificate is revoked. - iso15118CertificateHashData has a responder URL that points to an OCSP service for OCTT. - CSMS does not have a cached OCSP response for the contract certificate.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): State is *EVConnectedPreSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The OCTT sends an **AuthorizeRequest** With **idToken.idToken** *<Configured valid_idtoken_idtoken>* **idToken.type** *<Configured valid_idtoken_type>* **iso15118CertificateHashData** contains *<hashes from configured (V2G) certificate chain*	**2.** The CSMS sends an OCSP request to responder URL of **iso15118CertificateHashData** to check validity
	3. The OCTT OCSP service reponds that certificate is valid.	**4.** The CSMS responds with a **AuthorizeResponse**

Table 498. Test Case Id: TC_C_52_CSMS

Test case name	**Authorization using Contract Certificates 15118 - Online - Central contract certificate validation - Accepted**	
Test case Id	TC_C_52_CSMS	
Use case Id(s)	C07	
Requirement(s)	C07.FR.04,C07.FR.05	
System under test	CSMS	
Description	The Charging Station is able to authorize with contract certificates when it supports ISO 15118.	
Purpose	To verify if the CSMS is able to validate the provided certificate and eMAID. The field **iso15118CertificateHashData** is not provided to force CSMS to calculate certificate hash data for the OCSP request.	
Prerequisite(s)	- The configured eMAID is known by the CSMS as valid. - The configured contract certificate is signed by the configured V2GRoot or MORoot certificate at the CSMS. - Contract certificate has a responder URL that points to an OCSP service for OCTT. - CSMS does not have a cached OCSP response for the contract certificate.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): State is *EVConnectedPreSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The OCTT sends an **AuthorizeRequest** With **idToken.idToken** *<Configured valid_idtoken_idtoken>* **idToken.type** *<Configured valid_idtoken_type>* **iso15118CertificateHashData** is absent **certificate** is *<Configured contract_certificate>*	**2.** The CSMS sends an OCSP request to responder URL of **certificate** to check validity
	3. The OCTT OCSP service reponds that certificate is valid.	**4.** The CSMS responds with a **AuthorizeResponse**
	5. The OCTT sends a **TransactionEventRequest** With **triggerReason** *Authorized*	**6.** The CSMS responds with a **TransactionEventResponse**
	5. Execute **Reusable State** *EnergyTransferStarted*	
Tool validations	* Step 2: CSMS sends an OCSP request for **certificate** * Step 3: OCTT checks that received request for **certificate** is valid * Step 4: Message: **AuthorizeResponse** - **idTokenInfo.status** *Accepted* - **certificateStatus** *Accepted* * Step 6: Message: **TransactionEventResponse** - **idTokenInfo.status** *Accepted*	
	Post scenario validations: N/a	

3.5. D Local Authorization List Management

Table 499. Test Case Id: TC_D_01_CSMS

Test case name	**Send Local Authorization List - Full**	
Test case Id	TC_D_01_CSMS	
Use case Id(s)	D01	
Requirement(s)	D01_FR_01, D01_FR_06, D01_FR_18	
System under test	CSMS	
Description	The CSMS sends a Local Authorization List which a Charging Station can use for the authorization of idTokens. The list MAY be either a full list to replace the current list in the Charging Station or it MAY be a differential list with updates to be applied to the current list in the Charging Station.	
Purpose	To verify if the CSMS is able to send a Full Local Authorization List according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2 The OCTT responds with a **GetLocalListVersionResponse** with **versionNumber** *1*	**1.** The CSMS sends a **GetLocalListVersionRequest**
	Note(s): *This step is optional*	
	4 The OCTT responds with a **SendLocalListResponse** with **status** *Accepted*	**3.** The CSMS sends a **SendLocalListRequest**
	Note(s): *If the Local Authorization List is too big for one message, step 1 and 2 will be repeated*	
Tool validations	* Step 1: Message **SendLocalListRequest** - **updateType** *Full* - **versionNumber** *<Bigger than 0>* - **localAuthorizationList** *<Not empty>*	
	Post scenario validations: - N/a	

Table 500. Test Case Id: TC_D_02_CSMS

Test case name	**Send Local Authorization List - Differential Update**	
Test case Id	TC_D_02_CSMS	
Use case Id(s)	D01	
Requirement(s)	D01_FR_01, D01_FR_06, D01_FR_18	
System under test	CSMS	
Description	The CSMS sends a Local Authorization List which a Charging Station can use for the authorization of idTokens. The list MAY be either a full list to replace the current list in the Charging Station or it MAY be a differential list with updates to be applied to the current list in the Charging Station.	
Purpose	To verify if the CSMS is able to send a Differential Local Authorization List according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Request the CSMS to send a Local Authorization list to the Charging Station with type Differential and some idTokens in the message*	
	2 The OCTT responds with a **GetLocalListVersionResponse** with **versionNumber** *1*	**1.** The CSMS sends a **GetLocalListVersionRequest**
	4 The OCTT responds with a **SendLocalListResponse** with **status** *Accepted*	**3.** The CSMS sends a **SendLocalListRequest**
	Note(s): *If the Local Authorization List is too big for one message, step 1 and 2 will be repeated*	
Tool validations	* Step 1: Message **SendLocalListRequest** - **updateType** *Differential* - **versionNumber** *<Bigger than currently configured in OCTT>* - **localAuthorizationList** *<Not empty>*	
	Post scenario validations: - N/a	

Table 501. Test Case Id: TC_D_03_CSMS

Test case name	**Send Local Authorization List - Differential Remove**	
Test case Id	TC_D_03_CSMS	
Use case Id(s)	D01	
Requirement(s)	D01_FR_01, D01_FR_06, D01_FR_18, D01_FR_17	
System under test	CSMS	
Description	The CSMS sends a Local Authorization List which a Charging Station can use for the authorization of idTokens. The list MAY be either a full list to replace the current list in the Charging Station or it MAY be a differential list with updates to be applied to the current list in the Charging Station.	
Purpose	To verify if the CSMS is able to send a Differential Local Authorization List with data without idToken according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Request the CSMS to send a Local Authorization list to the Charging Station with type Differential and AuthorizationData elements without idTokenInfo in the message*	
	2 The OCTT responds with a **SendLocalListResponse** with **status** *Accepted*	**1.** The CSMS sends a **SendLocalListRequest**
	Note(s): *If the Local Authorization List is too big for one message, step 1 and 2 will be repeated*	
Tool validations	* Step 1: Message **SendLocalListRequest** - **updateType** *Differential* - **versionNumber** *<Bigger than currently configured in OCTT>* - **localAuthorizationList** *<AuthorizationData elements without idTokenInfo>*	
	Post scenario validations: - N/a	

Table 502. Test Case Id: TC_D_04_CSMS

Test case name	**Send Local Authorization List - Full with empy list**	
Test case Id	TC_D_04_CSMS	
Use case Id(s)	D01	
Requirement(s)	D01_FR_01, D01_FR_06, D01_FR_18	
System under test	CSMS	
Description	The CSMS sends a Local Authorization List which a Charging Station can use for the authorization of idTokens. The list MAY be either a full list to replace the current list in the Charging Station or it MAY be a differential list with updates to be applied to the current list in the Charging Station.	
Purpose	To verify if the CSMS is able to send a Full Local Authorization List without data according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Request the CSMS to send a Local Authorization list to the Charging Station with type full and without AuthorizationData elements in the message*	
	2 The OCTT responds with a **SendLocalListResponse** with **status** *Accepted*	**1.** The CSMS sends a **SendLocalListRequest**
	Note(s): *If the Local Authorization List is too big for one message, step 1 and 2 will be repeated*	
Tool validations	* Step 1: Message **SendLocalListRequest** - **updateType** *Full* - **localAuthorizationList** *<Empty>*	
	Post scenario validations: - N/a	

Table 503. Test Case Id: TC_D_08_CSMS

Test case name	**Get Local List Version - Success**	
Test case Id	TC_D_08_CSMS	
Use case Id(s)	D02	
Requirement(s)	N/a	
System under test	CSMS	
Description	The CSMS can request a Charging Station for the version number of the Local Authorization List by sending a GetLocalListVersionRequest.	
Purpose	To verify if the CSMS is able to request the Local Authorization List version according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Request the CSMS to get a Local Authorization list version*	
	2 The OCTT responds with a **GetLocalListVersionResponse** with **versionNumber** *<Configured versionNumber>*	**1.** The CSMS sends a **GetLocalListVersionRequest**
Tool validations	- N/a	
	Post scenario validations: - N/a	

Table 504. Test Case Id: TC_D_09_CSMS

Test case name	**Get Local List Version - No list available**	
Test case Id	TC_D_09_CSMS	
Use case Id(s)	D02	
Requirement(s)	N/a	
System under test	CSMS	
Description	The CSMS can request a Charging Station for the version number of the Local Authorization List by sending a GetLocalListVersionRequest.	
Purpose	To verify if the CSMS is able to request the Local Authorization List version according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Request the CSMS to get a Local Authorization list version*	
	2 The OCTT responds with a **GetLocalListVersionResponse** with **versionNumber** *0*	**1.** The CSMS sends a **GetLocalListVersionRequest**
Tool validations	- N/a	
	Post scenario validations: - N/a	

3.6. E Transactions

Table 505. Test Case Id: TC_E_01_CSMS

Test case name	**Start transaction options - PowerPathClosed**	
Test case Id	TC_E_01_CSMS	
Use case Id(s)	E01(S5)	
Requirement(s)	E01.FR.05	
System under test	CSMS	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the start options.	
Purpose	To verify if the CSMS is able to handle a Charging Station that starts a transaction when the power path has been closed.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The OCTT sends an **AuthorizeRequest** With **idToken.idToken** *<Configured valid_idtoken_idtoken>* **idToken.type** *<Configured valid_idtoken_type>*	**2.** The CSMS responds with an **AuthorizeResponse**
	3. The OCTT notifies the CSMS about the status change of the connector. Message: **StatusNotificationRequest** - **connectorStatus** is *Occupied* Message: **NotifyEventRequest** - **trigger** is *Delta* - **actualValue** is *Occupied* - **component.name** is *Connector* - **variable.name** is *AvailabilityState*	**4.** The CSMS responds accordingly.
	5. The OCTT sends a **TransactionEventRequest** With **eventType** is *Started* **triggerReason** is *ChargingStateChanged* **idToken.idToken** *<Configured valid_idtoken_idtoken>* **idToken.type** *<Configured valid_idtoken_type>* **evse.id** is *<Configured evseId>* **evse.connectorId** is *<Configured connectorId>* **transactionInfo.chargingState** is *SuspendedEVSE*	**6.** The CSMS responds with a **TransactionEventResponse**
	7. The OCTT sends a **TransactionEventRequest** With **eventType** is *Updated* **triggerReason** is *ChargingStateChanged* **transactionInfo.chargingState** is *Charging*	**8.** The CSMS responds with a **TransactionEventResponse**

Test case name	**Start transaction options - PowerPathClosed**
Tool validations	* Step 2: Message: **AuthorizeResponse** - **idTokenInfo.status** must be *Accepted* * Step 6: Message: **TransactionEventResponse** - **idTokenInfo.status** must be *Accepted*
	Post scenario validations: N/a

Table 506. Test Case Id: TC_E_02_CSMS

Test case name	**Start transaction options - EnergyTransfer**	
Test case Id	TC_E_02_CSMS	
Use case Id(s)	E01(S6)	
Requirement(s)	E01.FR.06	
System under test	CSMS	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the start options.	
Purpose	To verify if the CSMS is able to handle a Charging Station that starts a transaction when the energy transfer starts.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The OCTT sends an **AuthorizeRequest** With **idToken.idToken** *<Configured valid_idtoken_idtoken>* **idToken.type** *<Configured valid_idtoken_type>*	**2.** The CSMS responds with an **AuthorizeResponse**
	3. The OCTT notifies the CSMS about the status change of the connector. Message: **StatusNotificationRequest** - **connectorStatus** is *Occupied* Message: **NotifyEventRequest** - **trigger** is *Delta* - **actualValue** is *Occupied* - **component.name** is *Connector* - **variable.name** is *AvailabilityState*	**4.** The CSMS responds accordingly.
	5. The OCTT sends a **TransactionEventRequest** With **eventType** is *Started* **triggerReason** is *ChargingStateChanged* **idToken.idToken** *<Configured valid_idtoken_idtoken>* **idToken.type** *<Configured valid_idtoken_type>* **evse.id** is *<Configured evseId>* **evse.connectorId** is *<Configured connectorId>* **transactionInfo.chargingState** is *Charging*	**6.** The CSMS responds with a **TransactionEventResponse**
Tool validations	* Step 2: Message: **AuthorizeResponse** - **idTokenInfo.status** must be *Accepted* * Step 6: Message: **TransactionEventResponse** - **idTokenInfo.status** must be *Accepted*	
	Post scenario validations: N/a	

Table 507. Test Case Id: TC_E_03_CSMS

Test case name	**Local start transaction - Cable plugin first - Success**	
Test case Id	TC_E_03_CSMS	
Use case Id(s)	E02	
Requirement(s)	E02.FR.02	
System under test	CSMS	
Description	OCPP 2.x.x allows an EV driver to either first connect the EV and EVSE OR present a form of identification. Both sequences will result in being able to charge.	
Purpose	To verify if the CSMS is able to handle a Charging Station that is able to start a charging session when the EV driver first connects the EV and EVSE, before authorization.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): State is *EVConnectedPreSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *Authorized*	
	2. Execute **Reusable State** *EnergyTransferStarted*	
Tool validations	N/a	
	Post scenario validations: N/a	

Table 508. Test Case Id: TC_E_04_CSMS

Test case name	**Local start transaction - Authorization first - Success**	
Test case Id	TC_E_04_CSMS	
Use case Id(s)	E03	
Requirement(s)	E03.FR.02	
System under test	CSMS	
Description	OCPP 2.x.x allows an EV driver to either first connect the EV and EVSE OR present a form of identification. Both sequences will result in being able to charge.	
Purpose	To verify if the CSMS is able to handle a Charging Station that is able to start a charging session when the EV driver first presents a form of identification, before connecting the EV and EVSE.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *Authorized*	
	2. Execute **Reusable State** *EnergyTransferStarted*	
Tool validations	N/a	
	Post scenario validations: N/a	

Table 509. Test Case Id: TC_E_39_CSMS

Test case name	**Stop transaction options - Deauthorized - timeout**	
Test case Id	TC_E_39_CSMS	
Use case Id(s)	E03, E06	
Requirement(s)	E03.FR.04, E03.FR.05, E06.FR.04	
System under test	CSMS	
Description	OCPP 2.x.x allows an EV driver to either first connect the EV and EVSE OR present a form of identification. Both sequences will result in being able to charge.	
Purpose	To verify if the CSMS is able to handle a Charging Station that deauthorizes the transaction after the **EVConnectionTimeout** has expired.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): State is *Authorized*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The OCTT sends a **TransactionEventRequest** With **triggerReason** is *EVConnectTimeout* **transactionInfo.stoppedReason** is *Timeout* **eventType** is *Ended* Note(s): - *This step will be executed after the _<Configured EV connection timeout>* expires._	**2.** The CSMS responds with a **TransactionEventResponse**
Tool validations	N/a	
	Post scenario validations: N/a	

Table 510. Test Case Id: TC_E_14_CSMS

Test case name	**Stop transaction options - EVDisconnected - Charging Station side**	
Test case Id	TC_E_14_CSMS	
Use case Id(s)	E06(S2)	
Requirement(s)	E06.FR.02	
System under test	CSMS	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the stop options.	
Purpose	To verify if the CSMS is able to handle a Charging Station that stops a transaction when the EV and EVSE are disconnected at the Charging Station side.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): State is *EVConnectedPostSession*	
Main (Scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *EVDisconnected*	
Tool validations	N/a	
	Post scenario validations: N/a	

Table 511. Test Case Id: TC_E_20_CSMS

Test case name	**Stop transaction options - EVDisconnected - EV side (able to charge IEC 61851-1 EV)**	
Test case Id	TC_E_20_CSMS	
Use case Id(s)	E06(S2), E10	
Requirement(s)	E06.FR.02	
System under test	CSMS	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the stop options.	
Purpose	To verify if the CSMS is able to handle a Charging Station that stops a transaction when the EV and EVSE are disconnected at the EV side.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): State is *EnergyTransferSuspended*	
Main (Scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *EVDisconnected*	
Tool validations	N/a	
	Post scenario validations: N/a	

Table 512. Test Case Id: TC_E_15_CSMS

Test case name	**Stop transaction options - StopAuthorized - Local**	
Test case Id	TC_E_15_CSMS	
Use case Id(s)	E06(S3)	
Requirement(s)	E06.FR.03	
System under test	CSMS	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the stop options.	
Purpose	To verify if the CSMS is able to handle a Charging Station that stops a transaction when the EV driver locally stops the transaction.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): State is *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The OCTT sends a **TransactionEventRequest** With **triggerReason** is *StopAuthorized* **transactionInfo.stoppedReason** is *Local* **eventType** is *Ended*	**2.** The CSMS responds with a **TransactionEventResponse**
Tool validations	N/a	
	Post scenario validations: N/a	

Table 513. Test Case Id: TC_E_21_CSMS

Test case name	**Stop transaction options - StopAuthorized - Remote**	
Test case Id	TC_E_21_CSMS	
Use case Id(s)	E06(S3) AND F03	
Requirement(s)	E06.FR.03,F03.FR.01,F03.FR.09, F03.FR.10	
System under test	CSMS	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the stop options.	
Purpose	To verify if the CSMS is able to handle a Charging Station that stops a transaction when it receives a RequestStopTransactionRequest.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): **State** is *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Trigger the CSMS to request the Charging Station to stop the ongoing transaction.*	
	2. The OCTT responds with a **RequestStopTransactionResponse** with **status** *Accepted*	**1.** The CSMS sends a **RequestStopTransactionRequest**
	3. The OCTT sends a **TransactionEventRequest**. with **triggerReason** is *RemoteStop* **transactionInfo.stoppedReason** is *Remote* **eventType** is *Ended*	**4.** The CSMS responds with a **TransactionEventResponse**.
Tool validations	* Step 1: Message: **RequestStopTransactionRequest** - **transactionId** must equal *<transactionId provided by the OCTT in before state.>*	
	Post scenario validations: N/a	

Table 514. Test Case Id: TC_E_09_CSMS

Test case name	**Start transaction options - EVConnected**	
Test case Id	TC_E_09_CSMS	
Use case Id(s)	E01(S2)	
Requirement(s)	E01.FR.02	
System under test	CSMS	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the start options.	
Purpose	To verify if the CSMS is able to handle a Charging Station that starts a transaction when the EV and EVSE are connected.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The OCTT notifies the CSMS about the status change of the connector. Message: **StatusNotificationRequest** - **connectorStatus** is *Occupied* Message: **NotifyEventRequest** - **trigger** is *Delta* - **actualValue** is *Occupied* - **component.name** is *Connector* - **variable.name** is *AvailabilityState*	**2.** The CSMS responds accordingly.
	3. The OCTT sends a **TransactionEventRequest** With **eventType** is *Started* **triggerReason** is *CablePluggedIn* **evse.id** is *<Configured evseId>* **evse.connectorId** is *<Configured connectorId>* **transactionInfo.chargingState** is *EVConnected*	**4.** The CSMS responds with a **TransactionEventResponse**
Tool validations	N/a	
	Post scenario validations: N/a	

Table 515. Test Case Id: TC_E_10_CSMS

Test case name	**Start transaction options - Authorized - Local**	
Test case Id	TC_E_10_CSMS	
Use case Id(s)	E01(S3)	
Requirement(s)	E01.FR.03	
System under test	CSMS	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the start options.	
Purpose	To verify if the CSMS is able to handle a Charging Station that starts a transaction when the EV and EVSE are connected.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The OCTT sends an **AuthorizeRequest** With **idToken.idToken** *<Configured valid_idtoken_idtoken>* **idToken.type** *<Configured valid_idtoken_type>*	**2.** The CSMS responds with an **AuthorizeResponse**
	3. The OCTT sends a **TransactionEventRequest** With **eventType** is *Started* **triggerReason** is *Authorized* **idToken.idToken** *<Configured valid_idtoken_idtoken>* **idToken.type** *<Configured valid_idtoken_type>*	**4.** The CSMS responds with a **TransactionEventResponse**
Tool validations	* Step 2: Message: **AuthorizeResponse** - **idTokenInfo.status** must be *Accepted* * Step 4: Message: **TransactionEventResponse** - **idTokenInfo.status** must be *Accepted*	
	Post scenario validations: N/a	

Table 516. Test Case Id: TC_E_11_CSMS

Test case name	**Start transaction options - DataSigned**	
Test case Id	TC_E_11_CSMS	
Use case Id(s)	E01(S4)	
Requirement(s)	E01.FR.04	
System under test	CSMS	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the start options.	
Purpose	To verify if the CSMS is able to handle a Charging Station that starts a transaction when the signed meter values are received.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The OCTT sends an **AuthorizeRequest** With **idToken.idToken** *<Configured valid_idtoken_idtoken>* **idToken.type** *<Configured valid_idtoken_type>*	**2.** The CSMS responds with an **AuthorizeResponse**
	3. The OCTT notifies the CSMS about the status change of the connector. Message: **StatusNotificationRequest** - **connectorStatus** is *Occupied* Message: **NotifyEventRequest** - **trigger** is *Delta* - **actualValue** is *Occupied* - **component.name** is *Connector* - **variable.name** is *AvailabilityState*	**4.** The CSMS responds accordingly.
	5. The OCTT sends a **TransactionEventRequest** With **eventType** is *Started* **triggerReason** is *SignedDataReceived* **idToken.idToken** *<Configured valid_idtoken_idtoken>* **idToken.type** *<Configured valid_idtoken_type>* **evse.id** is *<Configured evseId>* **evse.connectorId** is *<Configured connectorId>* **meterValue** is provided with the following values: **sampledValue.value** is *0.0* **sampledValue.context** is *Transaction.Begin* **sampledValue.signedMeterValue** is *<Generated SignedMeterValueType>*	**6.** The CSMS responds with a **TransactionEventResponse**
	7. The OCTT sends a **TransactionEventRequest** With **eventType** is *Updated* **triggerReason** is *ChargingStateChanged* **transactionInfo.chargingState** is *Charging*	**8.** The CSMS responds with a **TransactionEventResponse**

Test case name	**Start transaction options - DataSigned**
Tool validations	* Step 2: Message: **AuthorizeResponse** - **idTokenInfo.status** must be *Accepted* * Step 6: Message: **TransactionEventResponse** - **idTokenInfo.status** must be *Accepted*
	Post scenario validations: N/a

Table 517. Test Case Id: TC_E_12_CSMS

Test case name	**Start transaction options - ParkingBayOccupied**	
Test case Id	TC_E_12_CSMS	
Use case Id(s)	E01(S1)	
Requirement(s)	E01.FR.01	
System under test	CSMS	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the start options.	
Purpose	To verify if the CSMS is able to handle a Charging Station that starts a transaction when the EV and EVSE are connected.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Scenario)	**Charging Station**	**CSMS**
	1. The OCTT sends a **TransactionEventRequest** With **eventType** is *Started* **triggerReason** is *EVDetected*	**2.** The CSMS responds with a **TransactionEventResponse**
Tool validations	N/a	
	Post scenario validations: N/a	

Table 518. Test Case Id: TC_E_38_CSMS

Test case name	**Local start transaction - EV not ready**	
Test case Id	TC_E_38_CSMS	
Use case Id(s)	E03	
Requirement(s)	N/a	
System under test	CSMS	
Description	OCPP 2.x.x allows an EV driver to either first connect the EV and EVSE OR present a form of identification. Both sequences will result in being able to charge.	
Purpose	To verify if the CSMS is able to handle a Charging Station that reports an EV is not ready to start the energy transfer (yet).	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): State is *Authorized*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *EVConnectedPreSession*	
	2. The OCTT sends a **TransactionEventRequest** With **triggerReason** is *ChargingStateChanged* **transactionInfo.chargingState** is *SuspendedEV* **eventType** is *Updated*	**3.** The CSMS responds with a **TransactionEventResponse**
Tool validations	N/a	
	Post scenario validations: N/a	

Table 519. Test Case Id: TC_E_07_CSMS

Test case name	**Stop transaction options - PowerPathClosed - Local stop**	
Test case Id	TC_E_07_CSMS	
Use case Id(s)	E06(S5)	
Requirement(s)	E06.FR.06	
System under test	CSMS	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the stop options.	
Purpose	To verify if the CSMS is able to handle a Charging Station that stops a transaction when it is locally stopped by an EV driver.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): State is *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The OCTT sends a **TransactionEventRequest** With **triggerReason** is *StopAuthorized* **transactionInfo.stoppedReason** is *Local* **eventType** is *Ended*	**2.** The CSMS responds with a **TransactionEventResponse**
Tool validations	N/a	
	Post scenario validations: N/a	

Table 520. Test Case Id: TC_E_08_CSMS

Test case name	**Stop transaction options - EnergyTransfer stopped - StopAuthorized**	
Test case Id	TC_E_08_CSMS	
Use case Id(s)	E06(S6)	
Requirement(s)	E06.FR.07	
System under test	CSMS	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the stop options.	
Purpose	To verify if the CSMS is able to handle a Charging Station that stops a transaction when the energy transfer stopped normally.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): State is *StopAuthorized*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The OCTT sends a **TransactionEventRequest** With **triggerReason** is *ChargingStateChanged* **transactionInfo.chargingState** is *EVConnected* **transactionInfo.stoppedReason** is *Local* **eventType** is *Ended*	**2.** The CSMS responds with a **TransactionEventResponse**
Tool validations	N/a	
	Post scenario validations: N/a	

Table 521. Test Case Id: TC_E_16_CSMS

Test case name	**Stop transaction options - Deauthorized - Invalid idToken**	
Test case Id	TC_E_16_CSMS	
Use case Id(s)	E06(S3)	
Requirement(s)	E06.FR.04,E01.FR.11,E01.FR.12	
System under test	CSMS	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the start options.	
Purpose	To verify if the CSMS is able to handle a Charging Station that stops a transaction when the transaction gets deauthorized by the status from the idTokenInfo at a TransactionEventResponse message and it has been configured to do so.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Scenario)	**Charging Station**	**CSMS**
	1. The OCTT sends a **TransactionEventRequest** With **triggerReason** is *Authorized* **idToken.idToken** *<Configured invalid_idtoken_idtoken>* **idToken.type** *<Configured invalid_idtoken_type>* **eventType** is *Started*	**2.** The CSMS responds with a **TransactionEventResponse**
	3. The OCTT sends a **TransactionEventRequest** With **eventType** *Ended* **triggerReason** *Deauthorized* **transactionInfo.stoppedReason** *DeAuthorized*	**4.** The CSMS responds with a **TransactionEventResponse**
Tool validations	* Step 2: Message: **TransactionEventResponse** - **idTokenInfo.status** must be *Invalid* or *Unknown+*	
	Post scenario validations: N/a	

Table 522. Test Case Id: TC_E_17_CSMS

Test case name	**Stop transaction options - Deauthorized - EV side disconnect**	
Test case Id	TC_E_17_CSMS	
Use case Id(s)	E06(S3)	
Requirement(s)	E06.FR.04	
System under test	CSMS	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the start options.	
Purpose	To verify if the CSMS is able to handle a Charging Station that stops a transaction when the transaction gets deauthorized by a connection loss from the EV side and it has been configured to do so.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): State is *EnergyTransferSuspended*	
Main (Scenario)	**Charging Station**	**CSMS**
	1. The OCTT sends a **TransactionEventRequest** **triggerReason** must be *EVCommunicationLost* **transactionInfo.chargingState** must be *Idle* **transactionInfo.stoppedReason** must be *EVDisconnected* **eventType** must be *Ended*	**2.** The CSMS responds with a **TransactionEventResponse**
Tool validations	N/a	
	Post scenario validations: N/a	

Table 523. Test Case Id: TC_E_22_CSMS

Test case name	**Stop transaction options - EnergyTransfer stopped - SuspendedEV**	
Test case Id	TC_E_22_CSMS	
Use case Id(s)	E06(S6)	
Requirement(s)	E06.FR.07	
System under test	CSMS	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the stop options.	
Purpose	To verify if the CSMS is able to handle a Charging Station that stops a transaction when the energy transfer stopped by the EV.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): State is *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The OCTT sends a **TransactionEventRequest** With **triggerReason** is *ChargingStateChanged* **transactionInfo.chargingState** is *SuspendedEV* **transactionInfo.stoppedReason** is *StoppedByEV* **eventType** is *Ended*	**2.** The CSMS responds with a **TransactionEventResponse**
Tool validations	N/a	
	Post scenario validations: N/a	

Table 524. Test Case Id: TC_E_19_CSMS

Test case name	**Stop transaction options - ParkingBayUnoccupied**	
Test case Id	TC_E_19_CSMS	
Use case Id(s)	E06(S1)	
Requirement(s)	E06.FR.01	
System under test	CSMS	
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the stop options.	
Purpose	To verify if the CSMS is able to handle a Charging Station that stops a transaction when the EV left the parking bay.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): State is *EVDisconnected*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The OCTT sends a **TransactionEventRequest** With **triggerReason** is *EVDeparted* **transactionInfo.stoppedReason** is *Local* **eventType** is *Ended*	**2.** The CSMS responds with a **TransactionEventResponse**
Tool validations	N/a	
	Post scenario validations: N/a	

Table 525. Test Case Id: TC_E_26_CSMS

Test case name	**Disconnect cable on EV-side - Suspend transaction**	
Test case Id	TC_E_26_CSMS	
Use case Id(s)	E10	
Requirement(s)	E10.FR.01	
System under test	CSMS	
Description	The Charging Station can behave in several different ways when the cable is disconnected at the EV side, based on its configuration. This test case tests one of the possible configuration settings.	
Purpose	To verify if the CSMS can handle a Charging Station that suspends the transaction when the EV and EVSE are disconnected at the EV side AND is able restart the energy transfer after reconnecting the EV and EVSE.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): State is *EnergyTransferSuspended*	
Main (Scenario)	**Charging Station**	**CSMS**
	1. The OCTT sends a **TransactionEventRequest** With **triggerReason** is *EVCommunicationLost* **transactionInfo.chargingState** is *Idle* **eventType** is *Updated*	**2.** The CSMS responds with a **TransactionEventResponse**
	3. The OCTT notifies the CSMS about the current state of the connector. Message: **StatusNotificationRequest** - **connectorStatus** *Available* - **evseId** *<Configured evseId>* - **connectorId** *<Configured connectorId>* Message: **NotifyEventRequest** - **trigger** *Delta* - **actualValue** *"Available"* - **component.name** *"Connector"* - **component.evse.id** *<Configured evseId>* - **component.evse.connectorid** *<Configured connectorId>* - **variable.name** *"AvailabilityState"*	**4.** The CSMS responds accordingly.
	5. The OCTT sends a **TransactionEventRequest** With **triggerReason** is *CablePluggedIn* **transactionInfo.chargingState** is *EVConnected* **eventType** is *Updated*	**6.** The CSMS responds with a **TransactionEventResponse**
	7. The OCTT sends a **TransactionEventRequest** With **triggerReason** is *ChargingStateChanged* **transactionInfo.chargingState** is *Charging* **eventType** is *Updated*	**8.** The CSMS responds with a **TransactionEventResponse**
Tool validations	N/a	
	Post scenario validations: N/a	

Table 526. Test Case Id: TC_E_29_CSMS

Test case name	**Check Transaction status - Transaction with id ongoing - with message in queue**	
Test case Id	TC_E_29_CSMS	
Use case Id(s)	E14	
Requirement(s)	E14.FR.02,E14.FR.04	
System under test	CSMS	
Description	The CSMS is able to request the status of a transaction and to find out whether there are queued transaction-related messages, using the **GetTransactionStatusRequest** message.	
Purpose	To verify if the CSMS is able to request the status of queued TransactionEventRequest messages from a specific transaction by sending a **GetTransactionStatusRequest** with a transactionId. The OCTT will respond that there are message(s) queued belonging to the ongoing transaction with the requested id.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): State is *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. *The OCTT closes the WebSocket connection.*	
	2. *The OCTT waits a number of seconds equal to* _*<Configured Transaction Duration>*, then it will reconnect to the CSMS._	
	4. The OCTT responds with a **GetTransactionStatusResponse** With **ongoingIndicator** is *true* **messagesInQueue** is *true*	**3.** The CSMS sends a **GetTransactionStatusRequest**
	5. The OCTT sends a **TransactionEventRequest** With **eventType** is *Updated* **meterValues** is present. **offline** is *true*	**6.** The CSMS responds with a **TransactionEventResponse**
Tool validations	* Step 3: Message: **GetTransactionStatusRequest** - **transactionId** *<Generated transactionId from Before>*	
	Post scenario validations: N/a	

Table 527. Test Case Id: TC_E_30_CSMS

Test case name	**Check Transaction status - Transaction with id ongoing - without message in queue**	
Test case Id	TC_E_30_CSMS	
Use case Id(s)	E14	
Requirement(s)	E14.FR.02,E14.FR.05	
System under test	CSMS	
Description	The CSMS is able to request the status of a transaction and to find out whether there are queued transaction-related messages, using the **GetTransactionStatusRequest** message.	
Purpose	To verify if the CSMS is able to request the status of queued TransactionEventRequest messages from a specific transaction by sending a **GetTransactionStatusRequest** with a transactionId. The OCTT will respond that there is NO message queued belonging to the ongoing transaction with the requested id.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): State is *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **GetTransactionStatusResponse** With **ongoingIndicator** is *true* **messagesInQueue** is *false*	**1.** The CSMS sends a **GetTransactionStatusRequest**
Tool validations	* Step 1: Message: **GetTransactionStatusRequest** - **transactionId** must be *<Generated transactionId from Before>*	
	Post scenario validations: N/a	

Table 528. Test Case Id: TC_E_31_CSMS

Test case name	**Check Transaction status - Transaction with id ended - with message in queue**	
Test case Id	TC_E_31_CSMS	
Use case Id(s)	E14	
Requirement(s)	E14.FR.03,E14.FR.04	
System under test	CSMS	
Description	The CSMS is able to request the status of a transaction and to find out whether there are queued transaction-related messages, using the **GetTransactionStatusRequest** message.	
Purpose	To verify if the CSMS is able to request the status of queued TransactionEventRequest messages from a specific transaction by sending a **GetTransactionStatusRequest** with a transactionId. The OCTT will respond that there are message(s) queued belonging to an ended transaction with the requested id.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): State is *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. *The OCTT closes the WebSocket connection.*	
	2. *The OCTT waits a number of seconds equal to* _*<Configured Transaction duration>*, then it will reconnect to the CSMS._	
	3. The OCTT sends a **StatusNotificationRequest** With **evseId** is *<Configured evseId>* **connectorId** is *<Configured connectorId>* **connectorStatus** is *Available*	**4.** The CSMS responds with a **StatusNotificationResponse**
	5. The OCTT sends a **TransactionEventRequest** With **eventType** is *Ended* **offline** is *true* **triggerReason** is *EVCommunicationLost* **transactionInfo.chargingState** is *Idle* **seqNo** *<Skips two sequence number values>*	**6.** The CSMS responds with a **TransactionEventResponse**
	8. The OCTT responds with a **GetTransactionStatusResponse** With **ongoingIndicator** is *false* **messagesInQueue** is *true*	**7.** The CSMS sends a **GetTransactionStatusRequest**
	9. The OCTT sends a **TransactionEventRequest** With **triggerReason** is *StopAuthorized* **eventType** is *Updated* **offline** is *true* **seqNo** *<This is the first of the two skipped values>*	**10.** The CSMS responds with a **TransactionEventResponse**

Test case name	Check Transaction status - Transaction with id ended - with message in queue	
11. The OCTT sends a **TransactionEventRequest** With **triggerReason** is *ChargingStateChanged* **transactionInfo.chargingState** is *EVConnected* **eventType** is *Updated* **offline** is *true* **seqNo** *<This is the second of the two skipped values>*	**12.** The CSMS responds with a **TransactionEventResponse**	**Tool validations**
* Step 5: Message: **GetTransactionStatusRequest** - **transactionId** *<Generated transactionId from Before>*		

Table 529. Test Case Id: TC_E_33_CSMS

Test case name	**Check Transaction status - Without transactionId - with message in queue**	
Test case Id	TC_E_33_CSMS	
Use case Id(s)	E14	
Requirement(s)	E14.FR.06,E14.FR.07	
System under test	CSMS	
Description	The CSMS is able to request the status of a transaction and to find out whether there are queued transaction-related messages, using the **GetTransactionStatusRequest** message.	
Purpose	To verify if the CSMS is able to request the status of queued TransactionEventRequest messages by sending a **GetTransactionStatusRequest** without a transactionId. The OCTT will respond that there are message(s) queued.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): State is *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. *The OCTT closes the WebSocket connection.*	
	2. *The OCTT waits a number of seconds equal to* _<*Configured Transaction Duration*>, then it will reconnect to the CSMS._	
	4. The OCTT responds with a **GetTransactionStatusResponse** With **ongoingIndicator** is omitted. **messagesInQueue** is *true*	**3.** The CSMS sends a **GetTransactionStatusRequest**
	5. The OCTT sends a **TransactionEventRequest** With **eventType** is *Updated* **meterValues** is present. **offline** is *true*	**6.** The CSMS responds with a **TransactionEventResponse**
Tool validations	* Step 3: Message: **GetTransactionStatusRequest** - **transactionId** must be omitted.	
	Post scenario validations: N/a	

Table 530. Test Case Id: TC_E_34_CSMS

Test case name	**Check Transaction status - Without transactionId - without message in queue**	
Test case Id	TC_E_34_CSMS	
Use case Id(s)	E14	
Requirement(s)	E14.FR.06,E14.FR.08	
System under test	CSMS	
Description	The CSMS is able to request the status of a transaction and to find out whether there are queued transaction-related messages, using the **GetTransactionStatusRequest** message.	
Purpose	To verify if the CSMS is able to request the status of queued TransactionEventRequest messages by sending a **GetTransactionStatusRequest** without a transactionId. The OCTT will respond that there are NO message(s) queued.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **GetTransactionStatusResponse** With **ongoingIndicator** is omitted. **messagesInQueue** is *false*	**1.** The CSMS sends a **GetTransactionStatusRequest**
Tool validations	* Step 1: Message: **GetTransactionStatusRequest** - **transactionId** must be omitted.	
	Post scenario validations: N/a	

Table 531. Test Case Id: TC_E_53_CSMS

Test case name	**Reset Sequence Number - CSMS accepting *seqNo* = 0 at start of transaction**	
Test case Id	TC_E_53_CSMS	
Use case Id(s)	E01	
Requirement(s)	E01.FR.07	
System under test	CSMS	
Description	OCPP 2.0.1 Edition 2 recommends that *seqNo* starts at 0 for every transaction. CSMS must therefore be robust to a *seqNo* that is not continuously increasing, but that restarts for new transactions. Since a TransactionEventRequest cannot be rejected, this can only be detected by either the complete absence of a TransactionEventResponse from CSMS or an otherwise misbehaving CSMS.	
Purpose	To verify if the CSMS accepts that a new transactions starts with a *seqNo* = 0.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *EnergyTransferStarted* Note(s): *New transaction will use seqNo 0 for the first TransactionEventRequest.*	
	2. Execute **Reusable State** *EVDisconnected*	
	3. Execute **Reusable State** *EnergyTransferStarted* Note(s): *New transaction will use seqNo 0 for the first TransactionEventRequest.*	
	4. Execute **Reusable State** *EVDisconnected*	
Tool validations	* Step 1: CSMS accepts the message **TransactionEventRequest** with *eventType* = `Started` and *seqNo* = 0 and answers with a **TransactionEventResponse** message.	
	* Step 3: CSMS accepts the message **TransactionEventRequest** with *eventType* = `Started` and *seqNo* = 0 and answers with a **TransactionEventResponse** message.	

3.7. F Remote Control

Table 532. Test Case Id: TC_F_01_CSMS

Test case name	**Remote start transaction - Cable plugin first**	
Test case Id	TC_F_01_CSMS	
Use case Id(s)	F01	
Requirement(s)	N/a	
System under test	CSMS	
Description	OCPP 2.x.x allows an EV driver to either first connect the EV and EVSE OR wait for/trigger a RequestStartTransactionRequest. Both sequences will result in being able to charge.	
Purpose	To verify if the CSMS is able to handle a Charging Station that is starts a charging session when the EV driver first connects the EV and EVSE, before receiving a RequestStartTransactionRequest message.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): State is *EVConnectedPreSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Trigger the CSMS to request the Charging Station to start a transaction.*	
	2. The OCTT responds with a **RequestStartTransactionResponse** with **status** *Accepted* **transactionId** is *<Generated transactionId>*	**1.** The CSMS sends a **RequestStartTransactionRequest**
	3. The OCTT sends a **TransactionEventRequest**. with **triggerReason** is *RemoteStart* **transactionInfo.remoteStartId** is *<By CSMS provided remoteStartID>* **eventType** is *Updated*	**4.** The CSMS responds with a **TransactionEventResponse**.
	5. Execute **Reusable State** *EnergyTransferStarted* (**State** is *Authorized* and _EVConnected = true)	
Tool validations	* Step 1: Message: **RequestStartTransactionRequest** - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>*	
	Post scenario validations: N/a	

Table 533. Test Case Id: TC_F_02_CSMS

Test case name	**Remote start transaction - Remote start first - AuthorizeRemoteStart is true**	
Test case Id	TC_F_02_CSMS	
Use case Id(s)	F02	
Requirement(s)	F02.FR.01, F01.FR.01	
System under test	CSMS	
Description	OCPP 2.x.x allows an EV driver to either first wait for/trigger a RequestStartTransactionRequest OR connect the EV and EVSE. Both sequences will result in being able to charge.	
Purpose	To verify if the CSMS is able to handle a Charging Station that starts a charging session when the Charging Stations receives a RequestStartTransactionRequest message (while AuthorizeRemoteStart is true), before the EV driver connects the EV and EVSE (within the connectionTimeout). The Charging Station has to authorize beforehand like a local action to start a transaction.	
Prerequisite(s)	**AuthEnabled** is NOT implemented with mutability ReadOnly and the value set to false	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	<u>Manual Action</u>: *Trigger the CSMS to request the Charging Station to start a transaction.*	
	2. The OCTT responds with a **RequestStartTransactionResponse** with **status** *Accepted* **transactionId** is omitted.	**1.** The CSMS sends a **RequestStartTransactionRequest**
	3. The OCTT sends a **AuthorizeRequest**. with **idToken.idToken** *<Configured valid_idtoken_idtoken>* **idToken.type** *<Configured valid_idtoken_type>*	**4.** The CSMS responds with a **AuthorizeResponse**.
	5. The OCTT sends a **TransactionEventRequest**. with **triggerReason** is *RemoteStart* **transactionInfo.remoteStartId** is *<By OCTT generated remoteStartID>* **eventType** is *Started*	**6.** The CSMS responds with a **TransactionEventResponse**.
	7. Execute **Reusable State** *EnergyTransferStarted* (**State** is *Authorized* and _EVConnected = false)	
Tool validations	* Step 1: Message: **RequestStartTransactionRequest** - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>* * Step 4: Message: **AuthorizeResponse** - **idTokenInfo.status** must be *Accepted*	
	Post scenario validations: N/a	

Table 534. Test Case Id: TC_F_03_CSMS

Test case name	**Remote start transaction - Remote start first - AuthorizeRemoteStart is false**	
Test case Id	TC_F_03_CSMS	
Use case Id(s)	F02	
Requirement(s)	F02.FR.01, F01.FR.02	
System under test	CSMS	
Description	OCPP 2.x.x allows an EV driver to either first wait for/trigger a RequestStartTransactionRequest OR connect the EV and EVSE. Both sequences will result in being able to charge.	
Purpose	To verify if the CSMS is able to handle a Charging Station that starts a charging session when the Charging Stations receives a RequestStartTransactionRequest message (while AuthorizeRemoteStart is false), before the EV driver connects the EV and EVSE (within the connectionTimeout). The Charging station does NOT have to authorize beforehand like a local action to start a transaction.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	<u>Manual Action</u>: *Trigger the CSMS to request the Charging Station to start a transaction.*	
	2. The OCTT responds with a **RequestStartTransactionResponse** with **status** *Accepted* **transactionId** is omitted.	**1.** The CSMS sends a **RequestStartTransactionRequest**
	3. The OCTT sends a **TransactionEventRequest**. with **triggerReason** is *RemoteStart* **transactionInfo.remoteStartId** is *<By OCTT generated remoteStartID>* **eventType** is *Started*	**4.** The CSMS responds with a **TransactionEventResponse**.
	5. Execute **Reusable State** *EnergyTransferStarted* (**State** is *Authorized* and _EVConnected = false)	
Tool validations	* Step 1: Message: **RequestStartTransactionRequest** - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>*	
	Post scenario validations: N/a	

Table 535. Test Case Id: TC_F_04_CSMS

Test case name	**Remote start transaction - Remote start first - Cable plugin timeout**	
Test case Id	TC_F_04_CSMS	
Use case Id(s)	F02, E03	
Requirement(s)	E03.FR.04, E03.FR.05	
System under test	CSMS	
Description	OCPP 2.x.x allows an EV driver to either first wait for/trigger a RequestStartTransactionRequest OR connect the EV and EVSE. Both sequences will result in being able to charge.	
Purpose	To verify if the CSMS is able to handle a Charging Station that deauthorizes the transaction after the **EVConnectionTimeout** has been reached.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	<u>Manual Action</u>: *Trigger the CSMS to request the Charging Station to start a transaction.*	
	2. The OCTT responds with a **RequestStartTransactionResponse** with **status** *Accepted* **transactionId** is omitted.	**1.** The CSMS sends a **RequestStartTransactionRequest**
	3. The OCTT sends a **TransactionEventRequest**. with **triggerReason** is *RemoteStart* **transactionInfo.remoteStartId** is *<By OCTT generated remoteStartID>* **eventType** is *Started*	**4.** The CSMS responds with a **TransactionEventResponse**.
	5. The OCTT sends a **TransactionEventRequest**. with **triggerReason** is *EVConnectTimeout* **eventType** is *Updated* <u>Note(s)</u>: *- This step will be executed after the _<Configured Transaction Duration>* has been reached._	**6.** The CSMS responds with a **TransactionEventResponse**.
Tool validations	* Step 1: Message: **RequestStartTransactionRequest** - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>*	
	Post scenario validations: N/a	

Table 536. Test Case Id: TC_F_06_CSMS

Test case name	**Remote unlock Connector - Without ongoing transaction - Accepted**	
Test case Id	TC_F_06_CSMS	
Use case Id(s)	F05	
Requirement(s)	n/a	
System under test	CSMS	
Description	This test case describes how the CSMS can be requested to sent an UnlockConnectorRequest to the charging station. It sometimes happens that a connector of a Charging Station socket does not unlock correctly. This happens most of the time when there is tension on the charging cable. This means the driver cannot unplug his charging cable from the Charging Station. To help a driver, the CSO can send a UnlockConnectorRequest to the Charging Station. The Charging Station will then try to unlock the connector again.	
Purpose	To verify if the CSMS is able to perform the remote unlock connector mechanism as described at the OCPP specification.	
Prerequisite(s)		
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **UnlockConnectorResponse** with **status** *Unlocked*	**1.** The CSMS sends a **UnlockConnectorRequest**
Tool validations	* Step 1: Message **UnlockConnectorRequest** - **evseId** *<Configured evseId>* - **connectorId** *<Configured connectorId>*	
	Post scenario validations: - N/a	

Table 537. Test Case Id: TC_F_11_CSMS

Test case name	**Trigger message - MeterValues - Specific EVSE**	
Test case Id	TC_F_11_CSMS	
Use case Id(s)	F06	
Requirement(s)	F06.FR.01,F06.FR.02	
System under test	CSMS	
Description	The CSMS can request a Charging Station to send Charging Station-initiated messages. In the request the CSMS indicates which message it wishes to receive.	
Purpose	To verify if the CSMS is able to trigger the Charging Station to send a MeterValuesRequest for a specific EVSE, using a TriggerMessageRequest.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **TriggerMessageResponse** with status *Accepted*	**1.** The CSMS sends a **TriggerMessageRequest**
	3. The OCTT sends a **MeterValuesRequest** With **evseId** *<Configured evseId>* **meterValue[0].sampledValue.context** *Trigger*	**4.** The CSMS responds with a **MeterValuesResponse**
Tool validations	* Step 1: Message: **TriggerMessageRequest** - **requestedMessage** must be *MeterValues* - **evse.id** must be *<Configured evseId>*	
	Post scenario validations: N/a	

Table 538. Test Case Id: TC_F_12_CSMS

Test case name	**Trigger message - MeterValues - All EVSE**	
Test case Id	TC_F_12_CSMS	
Use case Id(s)	F06	
Requirement(s)	F06.FR.01	
System under test	CSMS	
Description	The CSMS can request a Charging Station to send Charging Station-initiated messages. In the request the CSMS indicates which message it wishes to receive.	
Purpose	To verify if the CSMS is able to trigger the Charging Station to send a MeterValuesRequest for all EVSE, using a TriggerMessageRequest.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **TriggerMessageResponse** with status *Accepted*	**1.** The CSMS sends a **TriggerMessageRequest**
	3. The OCTT sends a **MeterValuesRequest** With **evseId** omitted **meterValue[0].sampledValue.context** *Trigger* Note(s): *- This step will be executed for every EVSE.*	**4.** The CSMS responds with a **MeterValuesResponse**
Tool validations	* Step 1: Message: **TriggerMessageRequest** - **requestedMessage** must be *MeterValues*	
	Post scenario validations: N/a	

Table 539. Test Case Id: TC_F_13_CSMS

Test case name	**Trigger message - TransactionEvent - Specific EVSE**	
Test case Id	TC_F_13_CSMS	
Use case Id(s)	F06	
Requirement(s)	F06.FR.01,F06.FR.02	
System under test	CSMS	
Description	The CSMS can request a Charging Station to send Charging Station-initiated messages. In the request the CSMS indicates which message it wishes to receive.	
Purpose	To verify if the CSMS is able to trigger the Charging Station to send a TransactionEventRequest for a specific EVSE, using a TriggerMessageRequest.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): State is *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **TriggerMessageResponse** with status *Accepted*	**1.** The CSMS sends a **TriggerMessageRequest**
	3. The OCTT sends a **TransactionEventRequest** With **evse.id** *<Configured evseId>* **triggerReason** *Trigger* **transactionInfo.chargingState** *Charging* **meterValue** is present **meterValue[0].sampledValue.context** *Trigger*	**4.** The CSMS responds with a **TransactionEventResponse**
Tool validations	* Step 1: Message: **TriggerMessageRequest** - **requestedMessage** must be *TransactionEvent* - **evse.id** must be *<Configured evseId>*	
	Post scenario validations: N/a	

Table 540. Test Case Id: TC_F_14_CSMS

Test case name	**Trigger message - TransactionEvent - All EVSE**	
Test case Id	TC_F_14_CSMS	
Use case Id(s)	F06	
Requirement(s)	F06.FR.01	
System under test	CSMS	
Description	The CSMS can request a Charging Station to send Charging Station-initiated messages. In the request the CSMS indicates which message it wishes to receive.	
Purpose	To verify if the CSMS is able to trigger the Charging Station to send a TransactionEventRequest for all EVSE, using a TriggerMessageRequest.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): State is *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **TriggerMessageResponse** with status *Accepted*	**1.** The CSMS sends a **TriggerMessageRequest**
	3. The OCTT sends a **TransactionEventRequest** With **evse.id** omitted **triggerReason** *Trigger* **transactionInfo.chargingState** *Charging* **meterValue** is present **meterValue[0].sampledValue.context** *Trigger* Note(s): - *This step will be executed for every EVSE.*	**4.** The CSMS responds with a **TransactionEventResponse**
Tool validations	* Step 1: Message: **TriggerMessageRequest** - **requestedMessage** must be *TransactionEvent*	
	Post scenario validations: N/a	

Table 541. Test Case Id: TC_F_15_CSMS

Test case name	**Trigger message - LogStatusNotification - Idle**	
Test case Id	TC_F_15_CSMS	
Use case Id(s)	F06	
Requirement(s)	F06.FR.01	
System under test	CSMS	
Description	The CSMS can request a Charging Station to send Charging Station-initiated messages. In the request the CSMS indicates which message it wishes to receive.	
Purpose	To verify if the CSMS is able to trigger the Charging Station to send a LogStatusNotificationRequest, using a TriggerMessageRequest.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **TriggerMessageResponse** with status *Accepted*	**1.** The CSMS sends a **TriggerMessageRequest**
	3. The OCTT sends a **LogStatusNotificationRequest** with **status** *Idle*	**4.** The CSMS responds with a **LogStatusNotificationResponse**
Tool validations	* Step 1: Message: **TriggerMessageRequest** - **requestedMessage** must be *LogStatusNotification*	
	Post scenario validations: N/a	

Table 542. Test Case Id: TC_F_18_CSMS

Test case name	**Trigger message - FirmwareStatusNotification - Idle**	
Test case Id	TC_F_18_CSMS	
Use case Id(s)	F06	
Requirement(s)	F06.FR.01	
System under test	CSMS	
Description	The CSMS can request a Charging Station to send Charging Station-initiated messages. In the request the CSMS indicates which message it wishes to receive.	
Purpose	To verify if the CSMS is able to trigger the Charging Station to send a FirmwareStatusNotificationRequest, using a TriggerMessageRequest.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **TriggerMessageResponse** with status *Accepted*	**1.** The CSMS sends a **TriggerMessageRequest**
	3. The OCTT sends a **FirmwareStatusNotificationRequest** with **status** *Idle*	**4.** The CSMS responds with a **FirmwareStatusNotificationResponse**
Tool validations	* Step 1: Message: **TriggerMessageRequest** - **requestedMessage** must be *FirmwareStatusNotification*	
	Post scenario validations: N/a	

Table 543. Test Case Id: TC_F_20_CSMS

Test case name	**Trigger message - Heartbeat**	
Test case Id	TC_F_20_CSMS	
Use case Id(s)	F06	
Requirement(s)	F06.FR.01	
System under test	CSMS	
Description	The CSMS can request a Charging Station to send Charging Station-initiated messages. In the request the CSMS indicates which message it wishes to receive.	
Purpose	To verify if the CSMS is able to trigger the Charging Station to send a HeartbeatRequest, using a TriggerMessageRequest.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **TriggerMessageResponse** with status *Accepted*	**1.** The CSMS sends a **TriggerMessageRequest**
	3. The OCTT sends a **HeartbeatRequest**	**4.** The CSMS responds with a **HeartbeatResponse**
Tool validations	* Step 1: Message: **TriggerMessageRequest** - **requestedMessage** must be *Heartbeat*	
	Post scenario validations: N/a	

Table 544. Test Case Id: TC_F_23_CSMS

Test case name	**Trigger message - StatusNotification - Specific EVSE - Available**	
Test case Id	TC_F_23_CSMS	
Use case Id(s)	F06	
Requirement(s)	F06.FR.01,F06.FR.02,F06.FR.13	
System under test	CSMS	
Description	The CSMS can request a Charging Station to send Charging Station-initiated messages. In the request the CSMS indicates which message it wishes to receive.	
Purpose	To verify if the CSMS is able to trigger the Charging Station to send a StatusNotificationRequest for a specific available EVSE, using a TriggerMessageRequest.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **TriggerMessageResponse** with status *Accepted*	**1.** The CSMS sends a **TriggerMessageRequest**
	3. The OCTT notifies the CSMS about the current state of the connector. Message: **StatusNotificationRequest** - **connectorStatus** *Available* - **evseId** *<Configured evseId>* - **connectorId** *<Configured connectorId>* Message: **NotifyEventRequest** - **trigger** *Delta* - **actualValue** *"Available"* - **component.name** *"Connector"* - **component.evse.id** *<Configured evseId>* - **component.evse.connectorid** *<Configured connectorId>* - **variable.name** *"AvailabilityState"*	**4.** The CSMS responds accordingly.
Tool validations	* Step 1: Message: **TriggerMessageRequest** - **requestedMessage** must be *StatusNotification* - **evse.id** must be *<Configured evseId>*	
	Post scenario validations: N/a	

Table 545. Test Case Id: TC_F_24_CSMS

Test case name	**Trigger message - StatusNotification - Specific EVSE - Occupied**	
Test case Id	TC_F_24_CSMS	
Use case Id(s)	F06	
Requirement(s)	F06.FR.01,F06.FR.02,F06.FR.13	
System under test	CSMS	
Description	The CSMS can request a Charging Station to send Charging Station-initiated messages. In the request the CSMS indicates which message it wishes to receive.	
Purpose	To verify if the CSMS is able to trigger the Charging Station to send a StatusNotificationRequest for a specific occupied EVSE, using a TriggerMessageRequest.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The OCTT notifies the CSMS about the current state of the connector. Message: **StatusNotificationRequest** - **connectorStatus** *Occupied* - **evseId** *<Configured evseId>* - **connectorId** *<Configured connectorId>* Message: **NotifyEventRequest** - **trigger** *Delta* - **actualValue** *"Occupied"* - **component.name** *"Connector"* - **component.evse.id** *<Configured evseId>* - **component.evse.connectorid** *<Configured connectorId>* - **variable.name** *"AvailabilityState"*	**2.** The CSMS responds accordingly.
	4. The OCTT responds with a **TriggerMessageResponse** with status *Accepted*	**3.** The CSMS sends a **TriggerMessageRequest**
	5. The OCTT notifies the CSMS about the current state of the connector. Message: **StatusNotificationRequest** - **connectorStatus** *Occupied* - **evseId** *<Configured evseId>* - **connectorId** *<Configured connectorId>* Message: **NotifyEventRequest** - **trigger** *Delta* - **actualValue** *"Occupied"* - **component.name** *"Connector"* - **component.evse.id** *<Configured evseId>* - **component.evse.connectorid** *<Configured connectorId>* - **variable.name** *"AvailabilityState"*	**6.** The CSMS responds accordingly.

Test case name	**Trigger message - StatusNotification - Specific EVSE - Occupied**
Tool validations	* Step 1: Message: **TriggerMessageRequest** - **requestedMessage** must be *StatusNotification* - **evse.id** must be *<Configured evseId>*
	Post scenario validations: N/a

Table 546. Test Case Id: TC_F_27_CSMS

Test case name	**Trigger message - NotImplemented**	
Test case Id	TC_F_27_CSMS	
Use case Id(s)	F06	
Requirement(s)	F06.FR.08	
System under test	CSMS	
Description	The CSMS can request a Charging Station to send Charging Station-initiated messages. In the request the CSMS indicates which message it wishes to receive.	
Purpose	To verify if the CSMS is able to handle a Charging Station that does not support the requested message value from a TriggerMessageRequest.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **TriggerMessageResponse** with status *NotImplemented*	**1.** The CSMS sends a **TriggerMessageRequest**
Tool validations	N/a	
	Post scenario validations: N/a	

3.8. G Availability

Table 547. Test Case Id: TC_G_03_CSMS

Test case name	**Change Availability EVSE - Operative to inoperative**	
Test case Id	TC_G_03_CSMS	
Use case Id(s)	G03	
Requirement(s)	N/a	
System under test	CSMS	
Description	This test case covers how the CSMS requests the Charging Station to change the availability of one of the EVSEs from Operative to Inoperative. An EVSE is considered Operative in any status other than Faulted and Unavailable.	
Purpose	To verify if the CSMS is able to perform the change availability mechanism as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *Unavailable* for *<Configured evseId>*	
Tool validations	N/a	
	Post scenario validations: - N/a	

Table 548. Test Case Id: TC_G_04_CSMS

Test case name	**Change Availability EVSE - Inoperative to operative**	
Test case Id	TC_G_04_CSMS	
Use case Id(s)	G03	
Requirement(s)	N/a	
System under test	CSMS	
Description	This test case covers how the CSMS requests the Charging Station to change the availability of one of the EVSEs from Inoperative to Operative. An EVSE is considered Operative in any status other than Faulted and Unavailable.	
Purpose	To verify if the CSMS is able to perform the change availability mechanism as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: *Unavailable* for *<Configured evseId>*	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	<u>Manual Action</u>: *Request the CSMS to change the availability of an EVSE to Operative.*	
	2. The OCTT responds with a **ChangeAvailabilityResponse** with **status** *Accepted*	**1.** The CSMS sends a **ChangeAvailabilityRequest**
	3. The OCTT notifies the CSMS about the current state of all connectors belonging to the specified EVSE (and optionally also from the EVSE itself). Message: **StatusNotificationRequest** - **connectorStatus** *Available* - **evseId** *<Configured evseId>* Message: **NotifyEventRequest** - **trigger** *Delta* - **actualValue** *"Available"* - **component.name** *"EVSE" / Connector* - **component.evse.id** *<Configured evseId>* - **variable.name** *"AvailabilityState"*	**4.** The CSMS responds accordingly.
Tool validations	* Step 1: Message **ChangeAvailabilityRequest** - **operationalStatus** *Operative* - **evse.id** *<Configured evseId>* - **connectorId** *omit*	
	Post scenario validations: - N/a	

Table 549. Test Case Id: TC_G_05_CSMS

Test case name	**Change Availability Charging Station - Operative to inoperative**	
Test case Id	TC_G_05_CSMS	
Use case Id(s)	G04	
Requirement(s)	N/a	
System under test	CSMS	
Description	This test case describes how the CSMS requests the Charging Station to change the availability from operative to inoperative. A Charging Station is considered Operative when it is charging or ready for charging. A Charging Station is considered Inoperative when it does not allow any charging.	
Purpose	To verify if the CSMS is able to perform the change availability mechanism as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	<u>Manual Action</u>: *Request the CSMS to change the availability of the Charging Station to Inoperative.*	
	2. The OCTT responds with a **ChangeAvailabilityResponse** with **status** *Accepted*	**1.** The CSMS sends a **ChangeAvailabilityRequest**
	3. The OCTT notifies the CSMS about the current state of all connectors Message: **StatusNotificationRequest** - **connectorStatus** *Unavailable* Message: **NotifyEventRequest** - **trigger** *Delta* - **actualValue** *"Unavailable"* - **component.name** *"Connector"* - **variable.name** *"AvailabilityState"*	**4.** The CSMS responds accordingly.
Tool validations	* Step 1: Message **ChangeAvailabilityRequest** - **operationalStatus** *Inoperative* - **evseId** *omit* - **connectorId** *omit*	
	Post scenario validations: - N/a	

Table 550. Test Case Id: TC_G_06_CSMS

Test case name	**Change Availability Charging Station - Inoperative to operative**	
Test case Id	TC_G_06_CSMS	
Use case Id(s)	G04	
Requirement(s)	N/a	
System under test	CSMS	
Description	This test case describes how the CSMS requests the Charging Station to change the availability from inoperative to operative. A Charging Station is considered Operative when it is charging or ready for charging. A Charging Station is considered Inoperative when it does not allow any charging.	
Purpose	To verify if the CSMS is able to perform the change availability mechanism as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): Charging Station set to *Unavailable* (Original status was Available)	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Request the CSMS to change the availability of the Charging Station to Inoperative.*	
	2. The OCTT responds with a **ChangeAvailabilityResponse** with **status** *Accepted*	**1.** The CSMS sends a **ChangeAvailabilityRequest**
	3. The OCTT notifies the CSMS about the current state of all connectors. Message: **StatusNotificationRequest** - **connectorStatus** *Available* Message: **NotifyEventRequest** - **trigger** *Delta* - **actualValue** *"Available"* - **component.name** *"Connector"* - **variable.name** *"AvailabilityState"*	**4.** The CSMS responds accordingly.
Tool validations	* Step 1: Message **ChangeAvailabilityRequest** - **operationalStatus** *Operative* - **evseId** *omit* - **connectorId** *omit*	
	Post scenario validations: - N/a	

Table 551. Test Case Id: TC_G_07_CSMS

Test case name	**Change Availability Connector - Operative to inoperative**	
Test case Id	TC_G_07_CSMS	
Use case Id(s)	G03	
Requirement(s)	N/a	
System under test	CSMS	
Description	This test case covers how the CSMS requests the Charging Station to change the availability of one of the Connectors of one EVSE from Operative to Inoperative. A Connector is considered Operative in any status other than Faulted and Unavailable.	
Purpose	To verify if the CSMS is able to perform the change availability mechanism as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Request the CSMS to change the availability of a Connector to Inoperative.*	
	2. The OCTT responds with a **ChangeAvailabilityResponse** with **status** *Accepted*	**1.** The CSMS sends a **ChangeAvailabilityRequest**
	3. The OCTT notifies the CSMS about the current state of the connector. Message: **StatusNotificationRequest** - **connectorStatus** *Unavailable* - **evseId** *<Configured evseId>* - **connectorId** *<Configured connectorId>* Message: **NotifyEventRequest** - **trigger** *Delta* - **actualValue** *"Unavailable"* - **component.name** *"Connector"* - **component.evse.id** *<Configured evseId>* - **component.evse.connectorid** *<Configured connectorId>* - **variable.name** *"AvailabilityState"*	**4.** The CSMS responds accordingly.
Tool validations	* Step 1: Message **ChangeAvailabilityRequest** - **operationalStatus** *Inoperative* - **evse.id** *<Configured evseId>* - **evse.connectorId** *<Configured connectorId>*	
	Post scenario validations: N/a	

Table 552. Test Case Id: TC_G_08_CSMS

Test case name	**Change Availability Connector - Inoperative to operative**	
Test case Id	TC_G_08_CSMS	
Use case Id(s)	G03	
Requirement(s)	N/a	
System under test	CSMS	
Description	This test case covers how the CSMS requests the Charging Station to change the availability of one of the Connectors from one EVSE from Inoperative to Operative. A Connector is considered Operative in any status other than Faulted and Unavailable.	
Purpose	To verify if the CSMS is able to perform the change availability mechanism as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: *Unavailable* for *<Configured connectorId>*	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Request the CSMS to change the availability of a Connector to Operative.*	
	2. The OCTT responds with a **ChangeAvailabilityResponse** with **status** *Accepted*	**1.** The CSMS sends a **ChangeAvailabilityRequest**
	3. The OCTT notifies the CSMS about the current state of the connector. Message: **StatusNotificationRequest** - **connectorStatus** *Available* - **evseId** *<Configured evseId>* - **connectorId** *<Configured connectorId>* Message: **NotifyEventRequest** - **trigger** *Delta* - **actualValue** *"Available"* - **component.name** *"Connector"* - **component.evse.id** *<Configured evseId>* - **component.evse.connectorid** *<Configured connectorId>* - **variable.name** *"AvailabilityState"*	**4.** The CSMS responds accordingly.
Tool validations	* Step 1: Message **ChangeAvailabilityRequest** - **operationalStatus** *Operative* - **evse.id** *<Configured evseId>* - **evse.connectorId** *<Configured connectorId>*	
	Post scenario validations: N/a	

Table 553. Test Case Id: TC_G_11_CSMS

Test case name	**Change Availability EVSE - With ongoing transaction**	
Test case Id	TC_G_11_CSMS	
Use case Id(s)	G03	
Requirement(s)	N/a	
System under test	CSMS	
Description	This test case covers how the CSMS requests the Charging Station to change the availability of one of the EVSEs from Operative to Inoperative. An EVSE is considered Operative in any status other than Faulted and Unavailable.	
Purpose	To verify if the CSMS is able to send a change availability request during a transaction according to the mechanism as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State: State is *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	Note(s): *Request the CSMS to change the availablitiy to inoperative*	
	2. The OCTT responds with a **ChangeAvailabilityResponse** with **status** *Scheduled*	**1.** The CSMS sends a **ChangeAvailabilityRequest**
	Note(s): *Wait for <Configured Transaction Duration>*	
	3. Execute **Reusable State** *StopAuthorized*	
	4. Execute **Reusable State** *EVConnectedPostSession*	
	5. Execute **Reusable State** *EVDisconnected*	
	6. The OCTT notifies the CSMS about the current state of all connectors with Message: **StatusNotificationRequest** - **connectorStatus** *Unavailable* - **evseId** *<Configured evseId>* OR Message: **NotifyEventRequest** - **trigger** *Delta* - **actualValue** *"Unavailable"* - **component.name** *"Connector"* - **component.evse.id** *<Configured evseId>* - **variable.name** *"AvailabilityState"*	**7.** The CSMS responds accordingly.
Tool validations	* Step 1: Message **ChangeAvailabilityRequest** - **operationalStatus** *Inoperative* - **evse.id** *<Configured evseId>* - **connectorId** *omit*	
	Post scenario validations: - A respond to report the state of a connector has been received for all connectors.	

Table 554. Test Case Id: TC_G_14_CSMS

Test case name	**Change Availability Charging Station - With ongoing transaction**	
Test case Id	TC_G_14_CSMS	
Use case Id(s)	G04	
Requirement(s)	N/a	
System under test	CSMS	
Description	This test case covers how the CSMS requests the Charging Station to change the availability of one of the EVSEs from Operative to Inoperative. An EVSE is considered Operative in any status other than Faulted and Unavailable.	
Purpose	To verify if the CSMS is able to send a change availability request during a transaction according to the mechanism as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State: State is *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	Note(s): *Request the CSMS to change the availability of the station to inoperative*	
	2. The OCTT responds with a **ChangeAvailabilityResponse** with **status** *Scheduled*	**1.** The CSMS sends a **ChangeAvailabilityRequest**
	3. The OCTT notifies the CSMS about the current state of all unoccupied connectors with Message: **StatusNotificationRequest** - **connectorStatus** *Unavailable*	**4.** The CSMS responds accordingly.
	Note(s): *Wait for <Configured Transaction Duration>*	
	5. Execute **Reusable State** *StopAuthorized*	
	6. Execute **Reusable State** *EVConnectedPostSession*	
	7. Execute **Reusable State** *EVDisconnected*	
	8. The OCTT notifies the CSMS about the current state of the configured connector with Message: **StatusNotificationRequest** - **connectorStatus** *Unavailable*	**9.** The CSMS responds accordingly.
Tool validations	* Step 1: Message **ChangeAvailabilityRequest** - **operationalStatus** *Inoperative* - **evseId** *omit* - **connectorId** *omit*	
	Post scenario validations: - A respond to report the state of a connector has been received for all connectors.	

Table 555. Test Case Id: TC_G_17_CSMS

Test case name	**Change Availability Connector - With ongoing transaction**	
Test case Id	TC_G_17_CSMS	
Use case Id(s)	G03	
Requirement(s)	N/a	
System under test	CSMS	
Description	This test case covers how the CSMS requests the Charging Station to change the availability of one of the EVSEs from Operative to Inoperative. An EVSE is considered Operative in any status other than Faulted and Unavailable.	
Purpose	To verify if the CSMS is able to send a change availability request during a transaction according to the mechanism as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State: State is *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	Note(s): *Request the CSMS to change the availablitiy of one connector to inoperative*	
	2. The OCTT responds with a **ChangeAvailabilityResponse** with **status** *Scheduled*	**1.** The CSMS sends a **ChangeAvailabilityRequest**
	Note(s): *Wait for <Configured Transaction Duration>*	
	3. Execute **Reusable State** *StopAuthorized*	
	4. Execute **Reusable State** *EVConnectedPostSession*	
	5. Execute **Reusable State** *EVDisconnected*	
	6. The OCTT notifies the CSMS about the current state of all connectors with Message: **StatusNotificationRequest** - **connectorStatus** *Unavailable* - **evseId** *<Configured evseId>* - **connectorId** *<Configured connectorId>*	**7.** The CSMS responds accordingly.
Tool validations	* Step 1: Message **ChangeAvailabilityRequest** - **operationalStatus** *Inoperative* - **evse.id** *<Configured evseId>* - **evse.connectorId** *<Configured connectorId>*	
	Post scenario validations: - A respond to report the state of a connector has been received for all connectors.	

Table 556. Test Case Id: TC_G_20_CSMS

Test case name	**Connector status Notification - Lock Failure**	
Test case Id	TC_G_20_CSMS	
Use case Id(s)	G05	
Requirement(s)	G05.FR.03	
System under test	CSMS	
Description	This test case describes how the EV Driver is prevented from starting a charge session at the Charging Station while the Connector is not locked properly.	
Purpose	To verify if the CSMS responds on a notifyeventrequest as described at the OCPP specification.	
Prerequisite(s)	- N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The OCTT sends a **NotifyEventRequest** with - **eventData.trigger** *Delta* - **eventData.component.name** *"ConnectorPlugRetentionLock"* - **eventData.variable.name** *"Problem"* - **eventData.actualValue** *"true"*	**2.** The CSMS responds with a **NotifyEventResponse**
Tool validations	N/a	
	Post scenario validations: - N/a	

3.9. H Reservation

Table 557. Test Case Id: TC_H_01_CSMS

Test case name	**Reserve a specific EVSE - Accepted - Valid idToken**	
Test case Id	TC_H_01_CSMS	
Use case Id(s)	H01(S2), H03	
Requirement(s)	N/a	
System under test	CSMS	
Description	The CSMS is able to reserve a specific EVSE for a specific IdToken by sending a ReserveNowRequest containing an evseId.	
Purpose	To verify if the CSMS is able to request the Charging Station to reserve a specific EVSE, until the EV Driver with the specified IdToken arrives.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *Reserved* for *<Configured evseId>*	
Tool validations	N/a	
	Post scenario validations: N/a	

Table 558. Test Case Id: TC_H_07_CSMS

Test case name	**Reserve a specific EVSE - Reservation Ended / not used**	
Test case Id	TC_H_07_CSMS	
Use case Id(s)	H01(S2), H04	
Requirement(s)	N/a	
System under test	CSMS	
Description	The CSMS is able to reserve a specific EVSE for a specific IdToken by sending a ReserveNowRequest containing an evseId.	
Purpose	To verify if the CSMS is able to handle a reservation that is canceled by the Charging Station, because the EV driver did not arrive before the set **expiryDateTime** was reached.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Trigger the CSMS to send a ReserveNowRequest for a specific EVSE.*	
	2. The OCTT responds with a **ReserveNowResponse** With status *Accepted*	**1.** The CSMS sends a **ReserveNowRequest** with **expiryDateTime** *current time + <configured transaction duration>*
	3. The OCTT notifies the CSMS about the current state of the connector(s) of the configured EVSE Message: **StatusNotificationRequest** with **connectorStatus** *Reserved* Message: **NotifyEventRequest** with **trigger** *Delta* **actualValue** *"Reserved"* **component.name** *"Connector"* **variable.name** *"AvailabilityState"*	**4.** The CSMS responds accordingly.
	5. The OCTT notifies the CSMS about the current state of the connector(s) of the configured EVSE Message: **StatusNotificationRequest** with **connectorStatus** *Available* Message: **NotifyEventRequest** with **trigger** *Delta* **actualValue** *"Available"* **component.name** *"Connector"* **variable.name** *"AvailabilityState"* Note(s): *- The OCTT waits until the provided expiryDateTime from step 1 expires before executing this step.*	**6.** The CSMS responds accordingly.
	7. The OCTT sends a **ReservationStatusUpdateRequest** With reservationUpdateStatus *Expired* reservationId *<id received at step 1>*	**8.** The CSMS responds with a **ReservationStatusUpdateResponse**

Test case name	**Reserve a specific EVSE - Reservation Ended / not used**
Tool validations	* Step 1: Message: **ReserveNowRequest** - **evseId** must be *<Configured evseId>* - **connectorType** must be omitted - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>*
	Post scenario validations: N/a

Table 559. Test Case Id: TC_H_08_CSMS

Test case name	**Reserve an unspecified EVSE - Accepted**	
Test case Id	TC_H_08_CSMS	
Use case Id(s)	H01(S1), H03	
Requirement(s)	N/a	
System under test	CSMS	
Description	The CSMS is able to reserve an unspecified EVSE for a specific IdToken by sending a ReserveNowRequest without an evseId.	
Purpose	To verify if the CSMS is able to request the Charging Station to reserve an unspecified EVSE, until the EV Driver with the specified IdToken arrives.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Trigger the CSMS to send a ReserveNowRequest for an unspecified EVSE.*	
	2. The OCTT responds with a **ReserveNowResponse** with **status** *Accepted*	**1.** The CSMS sends a **ReserveNowRequest**
	3. The OCTT notifies the CSMS about the current state of the connector(s) of the configured EVSE Message: **StatusNotificationRequest** with **connectorStatus** *Reserved* Message: **NotifyEventRequest** with **trigger** *Delta* **actualValue** *"Reserved"* **component.name** *"Connector"* **variable.name** *"AvailabilityState"* Note(s): *- The OCTT will execute this step, if it is configured with only one EVSE.*	**4.** The CSMS responds accordingly.
Tool validations	* Step 1: Message: **ReserveNowRequest** - **evseId** must be omitted - **connectorType** must be omitted - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>*	
	Post scenario validations: N/a	

Table 560. Test Case Id: TC_H_14_CSMS

Test case name	**Reserve an unspecified EVSE - Amount of EVSEs available equals the amount of reservations**	
Test case Id	TC_H_14_CSMS	
Use case Id(s)	H01(S1)	
Requirement(s)	N/a	
System under test	CSMS	
Description	The CSMS is able to reserve an unspecified EVSE for a specific IdToken by sending a ReserveNowRequest without an evseId.	
Purpose	To verify if the CSMS is able to handle that the Charging Station sets all available EVSE to reserved, when the amount of EVSEs available equals the amount of reservations.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	<u>Manual Action</u>: *Trigger the CSMS to send a ReserveNowRequest for an unspecified EVSE.*	
	2. The OCTT responds with a **ReserveNowResponse** with **status** *Accepted*	**1.** The CSMS sends a **ReserveNowRequest** <u>Note(s)</u>: *- This step needs to executed time the amount of EVSE configured for the OCTT.*
	3. The OCTT notifies the CSMS about the current state of all connectors Message: **StatusNotificationRequest** with **connectorStatus** *Reserved* Message: **NotifyEventRequest** with **trigger** *Delta* **actualValue** *"Reserved"* **component.name** *"Connector"* **variable.name** *"AvailabilityState"* <u>Note(s)</u>: *- This step will be executed after the last ReserveNowRequest has been sent from step 1.*	**4.** The CSMS responds accordingly.
Tool validations	* Step 1: Message: **ReserveNowRequest** - **evseId** must be omitted - **connectorType** must be omitted - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>*	
	Post scenario validations: N/a	

Table 561. Test Case Id: TC_H_15_CSMS

Test case name	**Reserve a connector with a specific type - Success**	
Test case Id	TC_H_15_CSMS	
Use case Id(s)	H01(S3), H03	
Requirement(s)	N/a	
System under test	CSMS	
Description	The CSMS is able to reserve an EVSE with a connector with a specific type for a specific IdToken by sending a ReserveNowRequest with a connectorType.	
Purpose	To verify if the CSMS is able to request the Charging Station to reserve an EVSE with a connector with a specific type, until the EV Driver with the specified IdToken arrives.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	<u>Manual Action</u>: *Trigger the CSMS to send a ReserveNowRequest for a specific ConnectorType.*	
	2. The OCTT responds with a **ReserveNowResponse** With status *Accepted*	**1.** The CSMS sends a **ReserveNowRequest**
	3. The OCTT notifies the CSMS about the current state of the connector Message: **StatusNotificationRequest** with **connectorStatus** *Reserved* Message: **NotifyEventRequest** with **trigger** *Delta* **actualValue** *"Reserved"* **component.name** *"Connector"* **variable.name** *"AvailabilityState"*	**4.** The CSMS responds accordingly.
Tool validations	* Step 1: Message: **ReserveNowRequest** - **evseId** must be omitted - **connectorType** must be *<Configured connectorType>* - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>*	
	Post scenario validations: N/a	

Table 562. Test Case Id: TC_H_17_CSMS

Test case name	**Cancel reservation of an EVSE - Success**	
Test case Id	TC_H_17_CSMS	
Use case Id(s)	H02	
Requirement(s)	N/a	
System under test	CSMS	
Description	The CSMS is able to cancel a reservation by sending a **CancelReservationRequest** to the Charging Station.	
Purpose	To verify if the CSMS is able to request the Charging Station to cancel a reservation, by sending a **CancelReservationRequest**	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Trigger the CSMS to send a ReserveNowRequest for a specific EVSE.*	
	2. The OCTT responds with a **ReserveNowResponse** with **status** *Accepted*	**1.** The CSMS sends a **ReserveNowRequest**
	3. The OCTT notifies the CSMS about the current state of the connector(s) of the configured EVSE Message: **StatusNotificationRequest** with **connectorStatus** *Reserved* Message: **NotifyEventRequest** with **trigger** *Delta* **actualValue** *"Reserved"* **component.name** *"Connector"* **variable.name** *"AvailabilityState"*	**4.** The CSMS responds accordingly.
	Manual Action: *Trigger the CSMS to send a CancelReservationRequest for the reservation created at step 1.*	
	6. The OCTT responds with a **CancelReservationResponse** With status *Accepted*	**5.** The CSMS sends a **CancelReservationRequest**
	7. The OCTT notifies the CSMS about the current state of the connector(s) of the configured EVSE Message: **StatusNotificationRequest** with **connectorStatus** *Available* Message: **NotifyEventRequest** with **trigger** *Delta* **actualValue** *"Available"* **component.name** *"Connector"* **variable.name** *"AvailabilityState"*	**8.** The CSMS responds accordingly.

Test case name	**Cancel reservation of an EVSE - Success**
Tool validations	* Step 1: Message: **ReserveNowRequest** - **evseId** must be *<Configured evseId>* - **connectorType** must be omitted - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>* * Step 5: Message: **CancelReservationRequest** - **reservationId** must be equal to the **id** provided at step 1
	Post scenario validations: N/a

Table 563. Test Case Id: TC_H_19_CSMS

Test case name	**Reserve a specific EVSE - Use a reserved EVSE with GroupId**	
Test case Id	TC_H_19_CSMS	
Use case Id(s)	H01, H03	
Requirement(s)	N/a	
System under test	CSMS	
Description	The CSMS is able to reserve an EVSE for a specific group by sending a **ReserveNowRequest** containing a **groupIdToken**.	
Purpose	To verify if the CSMS is able to request the Charging Station create a reservation for a specific group, by sending a **ReserveNowRequest** with a **groupIdToken**	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	<u>Manual Action</u>: *Trigger the CSMS to send a ReserveNowRequest with a groupIdToken for a specific EVSE.*	
	2. The OCTT responds with a **ReserveNowResponse** With status *Accepted*	**1.** The CSMS sends a **ReserveNowRequest**
	3. The OCTT notifies the CSMS about the current state of the connector(s) of the configured EVSE Message: **StatusNotificationRequest** with **connectorStatus** *Reserved* Message: **NotifyEventRequest** with **trigger** *Delta* **actualValue** *"Reserved"* **component.name** *"Connector"* **variable.name** *"AvailabilityState"*	**4.** The CSMS responds accordingly.
Tool validations	* Step 1: Message: **ReserveNowRequest** - **evseId** must be *<Configured evseId>* - **connectorType** must be omitted - **groupIdToken** must be provided - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>*	
	Post scenario validations: N/a	

Table 564. Test Case Id: TC_H_20_CSMS

Test case name	**Charging Station cancels reservation when Faulted**	
Test case Id	TC_H_20_CSMS	
Use case Id(s)	H01	
Requirement(s)	N/a	
System under test	CSMS	
Description	The Charging Station will cancel reservations, when the EVSE specified for a reservation is set to an inoperative state.	
Purpose	To verify if the CSMS is able to handle it when the reservation is canceled when the availability state of the EVSE specified for the reservation is set to ***Faulted*** by the OCTT.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Trigger the CSMS to send a ReserveNowRequest for a specific EVSE.*	
	2. The OCTT responds with a **ReserveNowResponse** With status *Accepted*	**1.** The CSMS sends a **ReserveNowRequest**
	3. The OCTT notifies the CSMS about the current state of the connector(s) of the configured EVSE Message: **StatusNotificationRequest** with **connectorStatus** *Reserved* Message: **NotifyEventRequest** with **trigger** *Delta* **actualValue** *"Reserved"* **component.name** *"Connector"* **variable.name** *"AvailabilityState"*	**4.** The CSMS responds accordingly.
	5. The OCTT notifies the CSMS about the current state of the connector(s) of the configured EVSE Message: **StatusNotificationRequest** with **connectorStatus** *Faulted* Message: **NotifyEventRequest** with **trigger** *Delta* **actualValue** *"Faulted"* **component.name** *"Connector"* **variable.name** *"AvailabilityState"*	**6.** The CSMS responds accordingly.
	7. The OCTT sends a **ReservationStatusUpdateRequest** With reservationUpdateStatus *Removed* reservationId *<id received at step 1>*	**8.** The CSMS responds with a **ReservationStatusUpdateResponse**
Tool validations	* Step 1: Message: **ReserveNowRequest** - **evseId** must be *<Configured evseId>* - **connectorType** must be omitted - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>*	
	Post scenario validations: N/a	

Table 565. Test Case Id: TC_H_22_CSMS

Test case name	**Reserve a specific EVSE - Configured to Reject**	
Test case Id	TC_H_22_CSMS	
Use case Id(s)	H01	
Requirement(s)		
System under test	CSMS	
Description	The CSMS is able to reserve a specific EVSE for a specific IdToken by sending a ReserveNowRequest containing an evseId.	
Purpose	To verify if the CSMS is able to correctly read the respond from a charging station when it is configured not to accept reservations.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **ReserveNowResponse** with - **status** *Rejected*	**1.** The CSMS sends a **ReserveNowRequest**
Tool validations	N/a	
	Post scenario validations: N/a	

3.10. I Tariff and Cost

Table 566. Test Case Id: TC_I_01_CSMS

Test case name	**Show EV Driver running total cost during charging - costUpdatedRequest**	
Test case Id	TC_I_01_CSMS	
Use case Id(s)	I02	
Requirement(s)	I02.FR.01	
System under test	CSMS	
Description	While a transaction is ongoing, the driver wants to know how much the running total cost is, updated at a relevant interval.	
Purpose	To verify if the CSMS is able to correctly send the running total cost as described in the OCPP specification.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The OCTT sends an **AuthorizeRequest** with **idToken.idToken** *<Configured valid_idtoken_idtoken>* **idToken.type** *<Configured valid_idtoken_type>*	**2.** The CSMS responds with an **AuthorizeResponse**
	3. The OCTT sends a **TransactionEventRequest** with - **triggerReason** *Authorized* - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>* - **eventType** *Updated*	**4.** The CSMS responds with a **TransactionEventResponse**
	5. Execute **Reusable State** *EVConnectedPreSession*	
	6. Execute **Reusable State** *EnergyTransferStarted*	
	7. The OCTT sends a **TransactionEventRequest** With **triggerReason** is *MeterValuePeriodic* **eventType** is *Updated* **timestamp** *<The intervals between the timestamps of the received Meter Value messages equals the configured sampled Meter Values interval>.* **sampledValue.context** is *Sample.Periodic* Note(s): *_- This step will be executed every _<Configured sampled Meter Values interval>* *- The OCTT will end the testcase after two MeterValues.*	**8.** The OCTT responds with a **TransactionEventResponse**
	10. The OCTT responds with a **CostUpdatedResponse**	**9.** The CSMS sends a **CostUpdatedRequest** Note(s): *- This step will be executed after every TransactionEventResponse, if the message did not contain a totalCost.*

Test case name	**Show EV Driver running total cost during charging - costUpdatedRequest**
Tool validations	* Step 2: Message **AuthorizeResponse** - **idTokenInfo.status** *Accepted* * Step 4: Message **TransactionEventResponse** - **idTokenInfo.status** *Accepted* - **totalCost** *<Optional>* * Step 7: Message (Optional) **CostUpdatedRequest** - **transactionId** *<Generated TransactionId>*
	Post scenario validations: - N/a

Table 567. Test Case Id: TC_I_02_CSMS

Test case name	**Show EV Driver Final Total Cost After Charging**	
Test case Id	TC_I_02_CSMS	
Use case Id(s)	I03	
Requirement(s)	I03.FR.02	
System under test	CSMS	
Description	While a transaction is ongoing, the driver wants to know how much the running total cost is, updated at a relevant interval.	
Purpose	To verify if the CSMS is able to correctly send the total cost as described in the OCPP specification.	
Prerequisite(s)	- N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): state is *EVConnectedPostSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The OCTT notifies the CSMS about the current state of the configured connector. Message: **StatusNotificationRequest** - **connectorStatus** *Available* Message: **NotifyEventRequest** - **trigger** *Delta* - **actualValue** *"Available"* - **component.name** *"Connector"* - **variable.name** *"AvailabilityState"*	**2.** The CSMS responds accordingly.
	3. The OCTT sends a **TransactionEventRequest** with - **triggerReason** *EVCommunicationLost* - **eventType** *Ended* - **transactionInfo.chargingState** *Idle* - **transactionInfo.stoppedReason** *EVDisconnected*	**4.** The CSMS responds with a **TransactionEventResponse**
Tool validations	* Step 4: Message **TransactionEventResponse** - **totalCost** *<Not omitted>*	
	Post scenario validations: - N/a	

3.11. J MeterValues

Table 568. Test Case Id: TC_J_01_CSMS

Test case name	**Clock-aligned Meter Values - No transaction ongoing**	
Test case Id	TC_J_01_CSMS	
Use case Id(s)	J01	
Requirement(s)	J01.FR.18	
System under test	CSMS	
Description	The Charging Station samples the electrical meter or other sensor/transducer hardware to provide information about its Meter Values. Depending on configuration settings, the Charging Station will send Meter Values.	
Purpose	To verify if the CSMS is able to handle a Charging Station sending clock-aligned Meter Values, when there is no ongoing transaction.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The OCTT notifies the CSMS about its measured Meter Values. Message: **MeterValuesRequest** - **timestamp** *<The intervals between the timestamps of the received Meter Value messages equals the _<Configured clock-aligned Meter Values interval>.* - **sampledValue.context** is *Sample.Clock* Message: **NotifyEventRequest** - **timestamp** *<The intervals between the timestamps of the received Meter Value messages equals the _<Configured clock-aligned Meter Values interval>.* - **trigger** is *Periodic* - **component.name** is *FiscalMetering* Note(s): *- This step will be executed every _<Configured clock-aligned Meter Values interval>* *- This step will be executed for evseId=0 and all configured EVSE.* *- The OCTT will end the testcase after it has send three Meter Value messages.*	**2.** The CSMS responds accordingly.
Tool validations	N/a	
	Post scenario validations: N/a	

Table 569. Test Case Id: TC_J_02_CSMS

Test case name	**Clock-aligned Meter Values - Transaction ongoing**	
Test case Id	TC_J_02_CSMS	
Use case Id(s)	J01	
Requirement(s)	J01.FR.18	
System under test	CSMS	
Description	The Charging Station samples the electrical meter or other sensor/transducer hardware to provide information about its Meter Values. Depending on configuration settings, the Charging Station will send Meter Values.	
Purpose	To verify if the CSMS is able to handle a Charging Station sending clock-aligned Meter Values, when there is an ongoing transaction.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): State is *EnergyTransferStarted* for *<Configured evseId>*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The OCTT notifies the CSMS about its measured Meter Values. Message: **MeterValuesRequest** - **timestamp** *<The intervals between the timestamps of the received Meter Value messages equals the configured clock-aligned Meter Values interval>.* - **sampledValue.context** is *Sample.Clock* Message: **NotifyEventRequest** - **timestamp** *<The intervals between the timestamps of the received Meter Value messages equals the configured clock-aligned Meter Values interval>.* - **trigger** is *Periodic* - **component.name** is *FiscalMetering* Note(s): *- This step will be executed every _<Configured clock-aligned Meter Values interval>* *- This step will be executed for evseId=0 and all configured idle EVSE.*	**2.** The CSMS responds accordingly.
	3. The OCTT sends a **TransactionEventRequest** With **triggerReason** is *MeterValueClock* **eventType** is *Updated* **timestamp** *<The intervals between the timestamps of the received Meter Value messages equals the configured clock-aligned Meter Values interval>.* **sampledValue.context** is *Sample.Clock* Note(s): *- This step will be executed every _<Configured clock-aligned Meter Values interval>* *- The OCTT will end the testcase after the _<Configured transaction duration>* is reached._	**4.** The CSMS responds with a **TransactionEventResponse**

Test case name	**Clock-aligned Meter Values - Transaction ongoing**
Tool validations	N/a
	Post scenario validations: N/a

Table 570. Test Case Id: TC_J_03_CSMS

Test case name	**Clock-aligned Meter Values - EventType Ended**	
Test case Id	TC_J_03_CSMS	
Use case Id(s)	J01	
Requirement(s)	J01.FR.18	
System under test	CSMS	
Description	The Charging Station samples the electrical meter or other sensor/transducer hardware to provide information about its Meter Values. Depending on configuration settings, the Charging Station will send Meter Values.	
Purpose	To verify if the CSMS is able to handle a Charging Station sending clock-aligned Meter Values, when a transaction ends.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): State is *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *EVDisconnected* - The **TransactionEventRequest** containing eventType *Ended* contains the MeterValue field. - **timestamp** *<The intervals between the timestamps of the received Meter Value messages equals the configured value at configured clock-aligned Tx ended Meter Values interval>*. - **sampledValue.context** is *Sample.Clock* AND the last one has *Transaction.End* Note(s): - *This step will be executed after the _<Configured transaction duration>* is reached._ - *This causes the transaction to stop.*	
Tool validations	N/a	
	Post scenario validations: N/a	

Table 571. Test Case Id: TC_J_04_CSMS

Test case name	**Clock-aligned Meter Values - Signed**	
Test case Id	TC_J_04_CSMS	
Use case Id(s)	J01	
Requirement(s)	J01.FR.21	
System under test	CSMS	
Description	The Charging Station samples the electrical meter or other sensor/transducer hardware to provide information about its Meter Values. Depending on configuration settings, the Charging Station will send Meter Values.	
Purpose	To verify if the CSMS is able to handle a Charging Station sending clock-aligned Meter Values, when a transaction ends.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): State is *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *EVDisconnected* - The **TransactionEventRequest** containing eventType *Ended* contains the MeterValue field. - **timestamp** *<The intervals between the timestamps of the received Meter Value messages equals the configured value at configured clock-aligned Tx ended Meter Values interval>*. - **sampledValue.context** is *Sample.Clock* AND the last one has *Transaction.End* - **sampledValue.signedMeterValue** is *<Generated SignedMeterValueType>* Note(s): - *This step will be executed after the _<Configured transaction duration>* is reached._ - *This causes the transaction to stop.*	
Tool validations	N/a	
	Post scenario validations: N/a	

Table 572. Test Case Id: TC_J_07_CSMS

Test case name	**Sampled Meter Values - EventType Started - EVSE known**	
Test case Id	TC_J_07_CSMS	
Use case Id(s)	J02	
Requirement(s)	J02.FR.19	
System under test	CSMS	
Description	The Charging Station samples the electrical meter or other sensor/transducer hardware to provide information about its Meter Values. Depending on configuration settings, the Charging Station will send Meter Values.	
Purpose	To verify if the CSMS is able to handle a Charging Station sending start sampled Meter Values, when a transaction starts.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *EVConnectedPreSession* - The **TransactionEventRequest** contains the MeterValue field. - **sampledValue.context** is *Transaction.Begin*	
Tool validations	N/a	
	Post scenario validations: N/a	

Table 573. Test Case Id: TC_J_08_CSMS

Test case name	**Sampled Meter Values - Context Transaction.Begin - EVSE not known**	
Test case Id	TC_J_08_CSMS	
Use case Id(s)	J02	
Requirement(s)	J02.FR.19	
System under test	CSMS	
Description	The Charging Station samples the electrical meter or other sensor/transducer hardware to provide information about its Meter Values. Depending on configuration settings, the Charging Station will send Meter Values.	
Purpose	To verify if the CSMS is able to handle a Charging Station sending start sampled Meter Values, when a transaction starts.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *Authorized*	
	2. Execute **Reusable State** *EVConnectedPreSession* - The **TransactionEventRequest** contains the MeterValue field. - **sampledValue.context** is *Transaction.Begin*	
	3. Execute **Reusable State** *EnergyTransferStarted*	
Tool validations	N/a	
	Post scenario validations: N/a	

Table 574. Test Case Id: TC_J_09_CSMS

Test case name	**Sampled Meter Values - EventType Updated**	
Test case Id	TC_J_09_CSMS	
Use case Id(s)	J02	
Requirement(s)	J02.FR.19	
System under test	CSMS	
Description	The Charging Station samples the electrical meter or other sensor/transducer hardware to provide information about its Meter Values. Depending on configuration settings, the Charging Station will send Meter Values.	
Purpose	To verify if the CSMS is able to handle a Charging Station sending sampled Meter Values, when there is an ongoing transaction.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): State is *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The OCTT sends a **TransactionEventRequest** With **triggerReason** is *MeterValuePeriodic* **eventType** is *Updated* **timestamp** *<The intervals between the timestamps of the received Meter Value messages equals the configured sampled Meter Values interval>.* **sampledValue.context** is *Sample.Periodic* *Note(s):* *_- This step will be executed every _<Configured sampled Meter Values interval>* *- The OCTT will end the testcase after three MeterValues.*	**2.** The CSMS responds with a **TransactionEventResponse**
Tool validations	N/a	
	Post scenario validations: N/a	

Table 575. Test Case Id: TC_J_10_CSMS

Test case name	**Sampled Meter Values - EventType Ended**	
Test case Id	TC_J_10_CSMS	
Use case Id(s)	J02	
Requirement(s)	J02.FR.19	
System under test	CSMS	
Description	The Charging Station samples the electrical meter or other sensor/transducer hardware to provide information about its Meter Values. Depending on configuration settings, the Charging Station will send Meter Values.	
Purpose	To verify if the CSMS is able to handle a Charging Station sending sampled Meter Values, when a transaction ends.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): State is *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *EVDisconnected* - The **TransactionEventRequest** containing eventType *Ended* contains the MeterValue field. - **timestamp** *<The intervals between the timestamps of the received Meter Value messages equals the configured value at configured clock-aligned Tx ended Meter Values interval>*. - **sampledValue.context** is *Sample.Periodic* AND the last one has *Transaction.End* Note(s): - *This step will be executed after the _<Configured transaction duration>* is reached._ - *This causes the transaction to stop.*	
Tool validations	N/a	
	Post scenario validations: N/a	

Table 576. Test Case Id: TC_J_11_CSMS

Test case name	**Sampled Meter Values - Signed**	
Test case Id	TC_J_11_CSMS	
Use case Id(s)	J02	
Requirement(s)	J02.FR.21	
System under test	CSMS	
Description	The Charging Station samples the electrical meter or other sensor/transducer hardware to provide information about its Meter Values. Depending on configuration settings, the Charging Station will send Meter Values.	
Purpose	To verify if the CSMS is able to handle a Charging Station sending sampled Meter Values, when a transaction ends.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): State is *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *EVDisconnected* - The **TransactionEventRequest** containing eventType *Ended* contains the MeterValue field. - **timestamp** *<The intervals between the timestamps of the received Meter Value messages equals the configured value at configured clock-aligned Tx ended Meter Values interval>*. - **sampledValue.context** is *Sample.Periodic* AND the last one has *Transaction.End* - **sampledValue.signedMeterValue** is *<Generated SignedMeterValueType>* Note(s): - *This step will be executed after the* _*<Configured transaction duration>* is reached._ - *This causes the transaction to stop.*	
Tool validations	N/a	
	Post scenario validations: N/a	

3.12. K SmartCharging

Table 577. Test Case Id: TC_K_01_CSMS

Test case name	**Set Charging Profile - TxDefaultProfile - Specific EVSE**	
Test case Id	TC_K_01_CSMS	
Use case Id(s)	K01	
Requirement(s)	K01.FR.31	
System under test	CSMS	
Description	To enable the CSMS to influence the charging power or current drawn from a specific EVSE or the entire Charging Station over a period of time. The CSMS sends a SetChargingProfileRequest to the Charging Station to influence the power or current drawn by EVs. The CSMS calculates a ChargingSchedule to stay within certain limits, which MAY be imposed by any external system.	
Purpose	To verify if the CSMS is able to send a TxDefaultProfile charging profile for a specific EVSE as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **SetChargingProfileResponse** with **status** *Accepted*	**1.** The CSMS sends a **SetChargingProfileRequest** with - **chargingProfile.id** *<Configured chargingProfileId>*
Tool validations	* Step 1: Message **SetChargingProfileRequest** **evseId** *<Configured evseId>* AND **chargingProfile.stackLevel** *<Configured stackLevel>* AND **chargingProfile.chargingProfilePurpose** *TxDefaultProfile* AND **chargingProfile.chargingProfileKind** *Absolute* AND **chargingProfile.validFrom** *now* AND **chargingProfile.validTo** *now + <Configured Charging Schedule Duration>* AND **chargingProfile.chargingSchedule.startSchedule** *now* AND **chargingProfile.chargingSchedule.chargingRateUnit** *<Configured chargingRateUnit>* AND **chargingProfile.chargingSchedule.duration** *<Configured duration>* AND **chargingProfile.chargingSchedule.chargingSchedulePeriod.startPeriod** *<Configured startPeriod>* AND **chargingProfile.chargingSchedule.chargingSchedulePeriod.limit** *6.0* or *6000.0* AND **chargingProfile.chargingSchedule.chargingSchedulePeriod.numberPhases** *<Configured numberPhases>* where *<Configured numberPhases>* not *3* OR **chargingProfile.chargingSchedule.chargingSchedulePeriod.numberPhases** *<Configured numberPhases>* or *<omit>* where *<Configured numberPhases> 3*	
	Post scenario validations: - N/a	

Table 578. Test Case Id: TC_K_02_CSMS

Test case name	**Set Charging Profile - TxProfile without ongoing transaction on the specified EVSE**	
Test case Id	TC_K_02_CSMS	
Use case Id(s)	K01	
Requirement(s)	N/a	
System under test	CSMS	
Description	To enable the CSMS to influence the charging power or current drawn from a specific EVSE or the entire Charging Station over a period of time. The CSMS sends a SetChargingProfileRequest to the Charging Station to influence the power or current drawn by EVs. The CSMS calculates a ChargingSchedule to stay within certain limits, which MAY be imposed by any external system.	
Purpose	To verify if the CSMS is able to send a TxProfile and read the charger’s feedback while no transaction is ongoing for a specific EVSE as described at the OCPP specification.	
Prerequisite(s)	If the CSMS supports sending a TxProfile while there is no transaction ongoing.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **SetChargingProfileResponse** with **status** *Rejected*	**1.** The CSMS sends a **SetChargingProfileRequest** - **chargingProfile.id** *<Configured chargingProfileId>*
Tool validations	* Step 1: Message **SetChargingProfileRequest** - **evseId** *<Configured evseId>* AND - **chargingProfile.chargingProfilePurpose** *TxProfile* AND - **chargingProfile.stackLevel** *<Configured stackLevel>* AND - **chargingProfile.chargingProfileKind** *Relative* AND - **chargingProfile.chargingSchedule.chargingRateUnit** *<Configured chargingRateUnit>* AND - **chargingProfile.chargingSchedule.chargingSchedulePeriod.startPeriod** *0* AND - **chargingProfile.chargingSchedule.chargingSchedulePeriod.limit** *7.0* or *7000.0* AND - **chargingProfile.chargingSchedule.chargingSchedulePeriod.numberPhases** *<Configured numberPhases>* where *<Configured numberPhases>* not *3* OR - **chargingProfile.chargingSchedule.chargingSchedulePeriod.numberPhases** *<Configured numberPhases>* or *<omit>* where *<Configured numberPhases> 3*	
	Post scenario validations: - N/a	

Table 579. Test Case Id: TC_K_03_CSMS

Test case name	**Set Charging Profile - ChargingStationMaxProfile**	
Test case Id	TC_K_03_CSMS	
Use case Id(s)	K01	
Requirement(s)	K01.FR.31, K01.FR.38	
System under test	CSMS	
Description	To enable the CSMS to influence the charging power or current drawn from a specific EVSE or the entire Charging Station over a period of time. The CSMS sends a SetChargingProfileRequest to the Charging Station to influence the power or current drawn by EVs. The CSMS calculates a ChargingSchedule to stay within certain limits, which MAY be imposed by any external system.	
Purpose	To verify if the CSMS is able to send a ChargingStationMaxProfile charging profile as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **SetChargingProfileResponse** with **status** *Accepted*	**1.** The CSMS sends a **SetChargingProfileRequest** - **chargingProfile.id** *<Configured chargingProfileId>*
Tool validations	* Step 1: Message **SetChargingProfileRequest** **evseId** *0* AND **chargingProfile.stackLevel** *<Configured stackLevel>* AND **chargingProfile.chargingProfilePurpose** *ChargingStationMaxProfile_* AND **chargingProfile.chargingProfileKind** *Absolute* OR *Relative* **chargingProfile.chargingSchedule.chargingRateUnit** *<Configured ChargingRateUnit>* **chargingProfile.chargingSchedule.duration** *<Configured duration>* **chargingProfile.chargingSchedule.chargingSchedulePeriod.startPeriod** *0* **chargingProfile.chargingSchedule.chargingSchedulePeriod.limit** *8.0* or *8000.0* **chargingProfile.chargingSchedule.chargingSchedulePeriod.numberPhases** *<Configured numberPhases>* where *<Configured numberPhases>* not *3* OR **chargingProfile.chargingSchedule.chargingSchedulePeriod.numberPhases** *<Configured numberPhases>* or *<omit>* where *<Configured numberPhases>* *3* **chargingProfile.validFrom** *<Not omitted>* **chargingProfile.validTo** *<Not omitted>* **chargingProfile.chargingSchedule.startSchedule** *<Not omitted>*	
	Post scenario validations: - N/a	

Table 580. Test Case Id: TC_K_04_CSMS

Test case name	**Replace charging profile - With chargingProfileId**	
Test case Id	TC_K_04_CSMS	
Use case Id(s)	n/a	
Requirement(s)	n/a	
System under test	CSMS	
Description	To enable the CSMS to influence the charging power or current drawn from a specific EVSE or the entire Charging Station over a period of time. The CSMS sends a SetChargingProfileRequest to the Charging Station to influence the power or current drawn by EVs. The CSMS calculates a ChargingSchedule to stay within certain limits, which MAY be imposed by any external system.	
Purpose	To verify if the CSMS is able to replace a charging profile with the same ProfileKind, Purpose, and stackLevel, but a different limit.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **SetChargingProfileResponse** with **status** *Accepted*	**1.** The CSMS sends a **SetChargingProfileRequest** with **chargingProfile.chargingSchedule.chargingSchedulePeriod.limit** *8.0 or 8000.0*
	4. The OCTT responds with a **SetChargingProfileResponse** with **status** *Accepted*	**3.** The CSMS sends a **SetChargingProfileRequest** with **chargingProfile.chargingSchedule.chargingSchedulePeriod.limit** *6.0 or 6000.0*
Tool validations	* Step 3: Message **SetChargingProfileRequest** **chargingProfile.id** *<same id for both chargingProfiles>*	
	Post scenario validations: - N/a	

Table 581. Test Case Id: TC_K_05_CSMS

Test case name	**Clear Charging Profile - With chargingProfileId**	
Test case Id	TC_K_05_CSMS	
Use case Id(s)	K10	
Requirement(s)	K10.FR.02	
System under test	CSMS	
Description	If the CSMS wishes to clear some or all of the charging profiles that were previously sent to the Charging Station, then the CSMS sends a ClearChargingProfileRequest to the Charging Station.	
Purpose	To verify if the CSMS is able to request the charging station to clear a specific charging profile (not TxDefault) with only a chargingProfileId as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: CSMS sends a **GetChargingProfilesRequest** OCTT responds with a **GetChargingProfilesResponse** with status *Accepted* OCTT sends a **ReportChargingProfilesRequest** CSMS responds with a **ReportChargingProfilesResponse**	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **ClearChargingProfileResponse** with **status** *Accepted*	**1.** The CSMS sends a **ClearChargingProfileRequest** with **chargingProfileId** *<Generated chargingProfileId>* AND **chargingProfileCriteria** *omit*
Tool validations	N/a	
	Post scenario validations: - N/a	

Table 582. Test Case Id: TC_K_06_CSMS

Test case name	**Clear Charging Profile - With stackLevel/purpose combination for one profile**	
Test case Id	TC_K_06_CSMS	
Use case Id(s)	K10	
Requirement(s)	K10.FR.02	
System under test	CSMS	
Description	If the CSMS wishes to clear some or all of the charging profiles that were previously sent to the Charging Station, then the CSMS sends a ClearChargingProfileRequest to the Charging Station.	
Purpose	To verify if the CSMS is able to request the charging station to clear a specific charging profile with a stackLevel/purpose combination for a chargingProfileId as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **ClearChargingProfileResponse** with **status** *Accepted*	**1.** The CSMS sends a **ClearChargingProfileRequest** with **chargingProfilePurpose** *TxDefaultProfile* AND **evseId** *<Configured evseId>* AND **stackLevel** *<Configured stackLevel>*
Tool validations	* Step 1: Message **ClearChargingProfileRequest** **chargingProfileCriteria.chargingProfilePurpose** *TxDefaultProfile* AND **chargingProfileCriteria.stackLevel** *<Configured stackLevel>* AND **chargingProfileCriteria.evseId** *<Configured evseId>*	
	Post scenario validations: - N/a	

Table 583. Test Case Id: TC_K_08_CSMS

Test case name	**Clear Charging Profile - Without previous charging profile**	
Test case Id	TC_K_08_CSMS	
Use case Id(s)	K10	
Requirement(s)	N/a	
System under test	CSMS	
Description	If the CSMS wishes to clear some or all of the charging profiles that were previously sent to the Charging Station, then the CSMS sends a ClearChargingProfileRequest to the Charging Station.	
Purpose	To verify if the CSMS is able to request the charging station to clear a specific charging profile with a chargingProfileId and stackLevel/purpose combination while the Charging stations does not accept as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **ClearChargingProfileResponse** with **status** *Unknown*	**1.** The CSMS sends a **ClearChargingProfileRequest** with **chargingProfilePurpose** *TxDefaultProfile* AND **evseId** *<Configured evseId>* AND **stackLevel** *<Configured stackLevel>*
Tool validations	* Step 1: Message **ClearChargingProfileRequest** **chargingProfilePurpose** *TxDefaultProfile* AND **evseId** *<Configured evseId>* AND **stackLevel** *<Configured stackLevel>*	
	Post scenario validations: - N/a	

Table 584. Test Case Id: TC_K_10_CSMS

Test case name	**Set Charging Profile - TxDefaultProfile - All EVSE**	
Test case Id	TC_K_10_CSMS	
Use case Id(s)	K01	
Requirement(s)	K01.FR.31	
System under test	CSMS	
Description	To enable the CSMS to influence the charging power or current drawn from a specific EVSE or the entire Charging Station over a period of time. The CSMS sends a SetChargingProfileRequest to the Charging Station to influence the power or current drawn by EVs. The CSMS calculates a ChargingSchedule to stay within certain limits, which MAY be imposed by any external system.	
Purpose	To verify if the CSMS is able to send a TxDefaultProfile charging profile for all EVSE as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **SetChargingProfileResponse** with **status** *Accepted*	**1.** The CSMS sends a **SetChargingProfileRequest** with - **chargingProfile.id** *<Configured chargingProfileId>*
Tool validations	* Step 1: Message **SetChargingProfileRequest** **evseId** *0* AND **chargingProfile.stackLevel** *<Configured stackLevel>* AND **chargingProfile.chargingProfilePurpose** *TxDefaultProfile* AND **chargingProfile.chargingProfileKind** *Absolute* AND **chargingProfile.validFrom** *<Not omitted>* AND **chargingProfile.validTo** *<Not omitted>* AND **chargingProfile.chargingSchedule.startSchedule** *<Not omitted>* AND **chargingProfile.chargingSchedule.chargingRateUnit** *<Configured ChargingRateUnit>* AND **chargingProfile.chargingSchedule.chargingSchedulePeriod.startPeriod** *0* AND **chargingProfile.chargingSchedule.duration** *<Configured duration>* **chargingProfile.chargingSchedule.chargingSchedulePeriod.limit** *6.0* or *6000.0* AND **chargingProfile.chargingSchedule.chargingSchedulePeriod.numberPhases** *<Configured numberPhases>* where *<Configured numberPhases>* not *3* OR **chargingProfile.chargingSchedule.chargingSchedulePeriod.numberPhases** *<Configured numberPhases>* or *<omit>* where *<Configured numberPhases> 3*	
	Post scenario validations: - N/a	

Table 585. Test Case Id: TC_K_15_CSMS

Test case name	**Set Charging Profile - Not Supported**	
Test case Id	TC_K_15_CSMS	
Use case Id(s)	K01	
Requirement(s)	N/a	
System under test	CSMS	
Description	To enable the CSMS to influence the charging power or current drawn from a specific EVSE or the entire Charging Station over a period of time. The CSMS sends a SetChargingProfileRequest to the Charging Station to influence the power or current drawn by EVs. The CSMS calculates a ChargingSchedule to stay within certain limits, which MAY be imposed by any external system.	
Purpose	To verify if the CSMS is able to send a Profile, while the charging station does not support chargingprofiles, and read the response as described at the OCPP specification.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with RPC Framework: CALLERROR: NotSupported.	**1.** The CSMS sends a **SetChargingProfileRequest** with: **evseId** *<Configured evseId>* AND **chargingProfile.stackLevel** *<Configured stackLevel>* AND **chargingProfile.chargingProfilePurpose** *TxDefaultProfile* AND **chargingProfile.chargingProfileKind** *Absolute* AND **chargingProfile.validFrom** *<Not omitted>* AND **chargingProfile.validTo** *<Not omitted>* AND **chargingProfile.chargingSchedule.startSchedule** *<Not omitted>* AND **chargingProfile.chargingSchedule.chargingRateUnit** *<Configured ChargingRateUnit>* AND **chargingProfile.chargingSchedule.chargingSchedulePeriod.startPeriod** *0* AND **chargingProfile.chargingSchedule.duration** *<Configured duration>* **chargingProfile.chargingSchedule.chargingSchedulePeriod.limit** *6.0* or *6000.0* AND **chargingProfile.chargingSchedule.chargingSchedulePeriod.numberPhases** *<Configured numberPhases>*
Tool validations	- N/a	
	Post scenario validations: - N/a	

Table 586. Test Case Id: TC_K_19_CSMS

Test case name	**Set Charging Profile - ChargingProfileKind is Recurring**	
Test case Id	TC_K_19_CSMS	
Use case Id(s)	K01	
Requirement(s)	N/a	
System under test	CSMS	
Description	To enable the CSMS to influence the charging power or current drawn from a specific EVSE or the entire Charging Station over a period of time. The CSMS sends a SetChargingProfileRequest to the Charging Station to influence the power or current drawn by EVs. The CSMS calculates a ChargingSchedule to stay within certain limits, which MAY be imposed by any external system.	
Purpose	To verify if the CSMS is able to send a Profile with a recurrencyKind specified as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **SetChargingProfileResponse** with - **status** *Accepted*	**1.** The CSMS sends a **SetChargingProfileRequest**
Tool validations	* Step 1: Message **SetChargingProfileRequest** - **evseId** *<Configured evseId>* AND - **chargingProfile.stackLevel** *<Configured stackLevel>* AND - **chargingProfile.chargingProfilePurpose** *TxDefaultProfile* AND - **chargingProfile.chargingSchedule.chargingSchedulePeriod.startPeriod** *0* AND - **chargingProfile.chargingProfileKind** *Recurring* AND - **chargingProfile.recurrencyKind** *<Configured recurrencyKind>*	
	Post scenario validations: - N/a	

Table 587. Test Case Id: TC_K_29_CSMS

Test case name	**Get Charging Profile - EvseId 0**	
Test case Id	TC_K_29_CSMS	
Use case Id(s)	K09	
Requirement(s)	K09.FR.03	
System under test	CSMS	
Description	With the GetChargingProfilesRequest message the CSMS can ask a Charging Station to report all, or a subset of all the install Charging Profiles from the different possible sources. This can be used for some automatic smart charging control system, or for debug purposes by a CSO.	
Purpose	To verify if the CSMS is able to request charging profiles installed on the charging station itself and read in the reports as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **GetChargingProfilesResponse** with - **status** *Accepted*	**1.** The CSMS sends a **GetChargingProfilesRequest** with - **evseId** *0*
	3. The OCTT sends a **ReportChargingProfilesRequest** with - **requestId** *<Received requestId>*	**4.** The CSMS responds with a **ReportChargingProfilesResponse**
Tool validations	* Step 1: Message **GetChargingProfilesRequest** - **evseId** *0* AND - **chargingProfile.chargingProfilePurpose** *<Configured chargingProfilePurpose>*	
	Post scenario validations: - N/a	

Table 588. Test Case Id: TC_K_30_CSMS

Test case name	**Get Charging Profile - EvseId > 0**	
Test case Id	TC_K_30_CSMS	
Use case Id(s)	K09	
Requirement(s)	K09.FR.03	
System under test	CSMS	
Description	With the GetChargingProfilesRequest message the CSMS can ask a Charging Station to report all, or a subset of all the install Charging Profiles from the different possible sources. This can be used for some automatic smart charging control system, or for debug purposes by a CSO.	
Purpose	To verify if the CSMS is able to request charging profiles installed on a specific EVSE and read in the reports as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
		1. The CSMS sends a **GetChargingProfilesRequest**
	2. The OCTT responds with a **GetChargingProfilesResponse** with - **status** *Accepted*	
	3. The OCTT sends a **ReportChargingProfilesRequest** with - **requestId** *<Received requestId>*	**4.** The CSMS responds with a **ReportChargingProfilesResponse**
Tool validations	* Step 1: Message **GetChargingProfilesRequest** - **evseId** *<Configured evseId>*	
	Post scenario validations: - N/a	

Table 589. Test Case Id: TC_K_31_CSMS

Test case name	**Get Charging Profile - No EvseId**	
Test case Id	TC_K_31_CSMS	
Use case Id(s)	K09	
Requirement(s)	K09.FR.03	
System under test	CSMS	
Description	With the GetChargingProfilesRequest message the CSMS can ask a Charging Station to report all, or a subset of all the install Charging Profiles from the different possible sources. This can be used for some automatic smart charging control system, or for debug purposes by a CSO.	
Purpose	To verify if the CSMS is able to request all charging profiles installed on a charger and read in the reports as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **GetChargingProfilesResponse** with - **status** *Accepted*	**1.** The CSMS sends a **GetChargingProfilesRequest** with - **requestId** *<Received requestId>*
	3. The OCTT sends a **ReportChargingProfilesRequest** with - **requestId** *<Received requestId>* AND - **tbc** *true* AND - **evseId** *i*	**4.** The CSMS responds with a **ReportChargingProfilesResponse**
	Note(s): *- Step 3 and 4 are repeated for every evse*	
Tool validations	* Step 1: Message **GetChargingProfilesRequest** - **evseId** *omit*	
	Post scenario validations: - N/a	

Table 590. Test Case Id: TC_K_32_CSMS

Test case name	**Get Charging Profile - chargingProfileId**	
Test case Id	TC_K_32_CSMS	
Use case Id(s)	K09	
Requirement(s)	K09.FR.03	
System under test	CSMS	
Description	With the GetChargingProfilesRequest message the CSMS can ask a Charging Station to report all, or a subset of all the install Charging Profiles from the different possible sources. This can be used for some automatic smart charging control system, or for debug purposes by a CSO.	
Purpose	To verify if the CSMS is able to request a specific charging profile and read in the reports as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
		1. The CSMS sends a **GetChargingProfilesRequest** - **chargingProfileId** *<Received chargingProfileId>*
	2. The OCTT responds with a **GetChargingProfilesResponse** with - **status** *Accepted*	
	3. The OCTT sends a **ReportChargingProfilesRequest** with - **requestId** *Generated Id*	
		4. The CSMS responds with a **ReportChargingProfilesResponse**
Tool validations	* Step 1: Message **GetChargingProfilesRequest** - **chargingProfileId** *<received chargingProfileId>* AND - **requestId** *<Generated Id>*	
	Post scenario validations: - N/a	

Table 591. Test Case Id: TC_K_33_CSMS

Test case name	**Get Charging Profile - EvseId > 0 + stackLevel**	
Test case Id	TC_K_33_CSMS	
Use case Id(s)	K09	
Requirement(s)	K09.FR.03	
System under test	CSMS	
Description	With the GetChargingProfilesRequest message the CSMS can ask a Charging Station to report all, or a subset of all the install Charging Profiles from the different possible sources. This can be used for some automatic smart charging control system, or for debug purposes by a CSO.	
Purpose	To verify if the CSMS is able to request charging profiles with a specific stackLevel installed on a specific EVSE and read in the reports as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **GetChargingProfilesResponse** with - **status** *Accepted*	**1.** The CSMS sends a **GetChargingProfilesRequest**
	3. The OCTT sends a **ReportChargingProfilesRequest** with - **requestId** *Generated Id*	**4.** The CSMS responds with a **ReportChargingProfilesResponse**
Tool validations	* Step 1: Message **GetChargingProfilesRequest** - **evseId** *<Configured evseId>* AND - **chargingProfile.stackLevel** *<Configured stackLevel>*	
	Post scenario validations: - N/a	

Table 592. Test Case Id: TC_K_34_CSMS

Test case name	**Get Charging Profile - EvseId > 0 + chargingLimitSource**	
Test case Id	TC_K_34_CSMS	
Use case Id(s)	K09	
Requirement(s)	K09.FR.03	
System under test	CSMS	
Description	With the GetChargingProfilesRequest message the CSMS can ask a Charging Station to report all, or a subset of all the install Charging Profiles from the different possible sources. This can be used for some automatic smart charging control system, or for debug purposes by a CSO.	
Purpose	To verify if the CSMS is able to request charging profiles with a specific chargingLimitSource installed on a specific EVSE and read in the reports as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
		1. The CSMS sends a **GetChargingProfilesRequest**
	2. The OCTT responds with a **GetChargingProfilesResponse** with - **status** *Accepted*	
	3. The OCTT sends a **ReportChargingProfilesRequest** with - **requestId** *Generated Id*	**4.** The CSMS responds with a **ReportChargingProfilesResponse**
Tool validations	* Step 1: Message **GetChargingProfilesRequest** - **evseId** *<Configured evseId>* AND - **chargingProfile.chargingLimitSource** *<Configured chargingLimitSource>*	
	Post scenario validations: - N/a	

Table 593. Test Case Id: TC_K_35_CSMS

Test case name	**Get Charging Profile - EvseId > 0 + chargingProfilePurpose**	
Test case Id	TC_K_35_CSMS	
Use case Id(s)	K09	
Requirement(s)	K09.FR.03	
System under test	CSMS	
Description	With the GetChargingProfilesRequest message the CSMS can ask a Charging Station to report all, or a subset of all the install Charging Profiles from the different possible sources. This can be used for some automatic smart charging control system, or for debug purposes by a CSO.	
Purpose	To verify if the CSMS is able to request charging profiles with a specific chargingProfilePurpose installed on a specific EVSE and read in the reports as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **GetChargingProfilesResponse** with - **status** *Accepted*	**1.** The CSMS sends a **GetChargingProfilesRequest**
	3. The OCTT sends a **ReportChargingProfilesRequest** with - **requestId** *Generated Id*	**4.** The CSMS responds with a **ReportChargingProfilesResponse**
Tool validations	* Step 1: Message **GetChargingProfilesRequest** - **evseId** *<Configured evseId>* AND - **chargingProfile.chargingProfilePurpose** *<Configured chargingProfilePurpose>*	
	Post scenario validations: - N/a	

Table 594. Test Case Id: TC_K_36_CSMS

Test case name	**Get Charging Profile - EvseId > 0 + chargingProfilePurpose + stackLevel**	
Test case Id	TC_K_36_CSMS	
Use case Id(s)	K09	
Requirement(s)	K09.FR.03	
System under test	CSMS	
Description	With the GetChargingProfilesRequest message the CSMS can ask a Charging Station to report all, or a subset of all the install Charging Profiles from the different possible sources. This can be used for some automatic smart charging control system, or for debug purposes by a CSO.	
Purpose	To verify if the CSMS is able to request charging profiles with a specific chargingProfilePurpose AND stackLevel installed on a specific EVSE and read in the reports as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **GetChargingProfilesResponse** with - **status** *Accepted*	**1.** The CSMS sends a **GetChargingProfilesRequest**
	3. The OCTT sends a **ReportChargingProfilesRequest** with - **requestId** *Generated Id*	**4.** The CSMS responds with a **ReportChargingProfilesResponse**
Tool validations	* Step 1: Message **GetChargingProfilesRequest** - **evseId** *<Configured evseId>* AND - **chargingProfile.chargingProfilePurpose** *<Configured chargingProfilePurpose>* - **chargingProfile.stackLevel** *<Configured stackLevel>*	
	Post scenario validations: - N/a	

Table 595. Test Case Id: TC_K_60_CSMS

Test case name	**Set Charging Profile - TxProfile with ongoing transaction on the specified EVSE**	
Test case Id	TC_K_60_CSMS	
Use case Id(s)	K01	
Requirement(s)	K01.FR.03, K01.FR.31	
System under test	CSMS	
Description	The CSMS sets a TxProfile on a specific EVSE for a currently ongoing transaction.	
Purpose	To verify if the CSMS is able to exchange messages to set a TxProfile on a specific EVSE for a currently ongoing transaction.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): State is *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **SetChargingProfileResponse** With **status** is *Accepted*	**1.** The CSMS sends a **SetChargingProfileRequest**
Tool validations	* Step 1: (Message: **SetChargingProfileRequest**) **ChargingProfilePurpose** is *TxProfile* AND **evseId** is *<Configured evseId>* AND **transactionId** *<Generated transactionId>*	
	Post scenario validations: N/a	

Table 596. Test Case Id: TC_K_37_CSMS

Test case name	**Remote start transaction with charging profile - Success**	
Test case Id	TC_K_37_CSMS	
Use case Id(s)	K05,F01	
Requirement(s)	K05.FR.02,F01.FR.08,F01.FR.09,F01.FR.11	
System under test	CSMS	
Description	The CSMS sets a TxProfile on a specific EVSE inside a RequestStartTransactionRequest message.	
Purpose	To verify if the CSMS is able to set a TxProfile on a specific EVSE in a RequestStartTransactionRequest message.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **RequestStartTransactionResponse** With **status** *Accepted*	**1.** The CSMS sends a **RequestStartTransactionRequest**
	3. The OCTT sends a **TransactionEventRequest** With **triggerReason** *RemoteStart* **transactionInfo.remoteStartId** is present.	**4.** The CSMS responds with a **TransactionEventResponse**
Tool validations	* Step 1: Message: **RequestStartTransactionRequest** with **idToken.idToken** *<Configured valid_idtoken_idtoken>* **idToken.type** *<Configured valid_idtoken_type>* **idToken.idToken** *<Configured valid idToken>* **idToken.type** *<Configured valid idToken type>* **evseId** *<Configured evseId>* **chargingProfile** contains: **chargingProfile.chargingProfilePurpose** is *TxProfile* **chargingProfile.transactionId** is omitted **chargingProfile.chargingProfileKind** is *Relative*	
	Post scenario validations: N/a	

Table 597. Test Case Id: TC_K_43_CSMS

Test case name	**Get Composite Schedule - Specific EVSE**	
Test case Id	TC_K_43_CSMS	
Use case Id(s)	K08	
Requirement(s)	K08.FR.01	
System under test	CSMS	
Description	The CSMS requests a composite schedule which is a combination of local limits and the prevailing Charging Profiles of the different chargingProfilePurposes and stack levels.	
Purpose	To verify if the CSMS is able to calculate request a composite schedule from the Charging Station for a specific EVSE.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **GetCompositeScheduleResponse** With **status** *Accepted* **schedule.evseId** *1* **schedule.duration** is *300* **schedule.chargingRateUnit** *<Specified chargingRateUnit from step 1>* **schedule.chargingSchedulePeriod[0].startPeriod** *0* *Note: Multiply limit by 1000 if chargingRateUnit is W* **schedule.chargingSchedulePeriod[0].limit** *10*	**1.** The CSMS sends a **GetCompositeScheduleRequest**
Tool validations	* Step 1: (Message: **GetCompositeScheduleRequest**) **evseId** *1* **duration** is *<Configured duration>* **chargingRateUnit** *<Configured chargingRateUnit>*	
	Post scenario validations: N/a	

Table 598. Test Case Id: TC_K_44_CSMS

Test case name	**Get Composite Schedule - Charging Station**	
Test case Id	TC_K_44_CSMS	
Use case Id(s)	K08	
Requirement(s)	K08.FR.01	
System under test	CSMS	
Description	The CSMS requests a composite schedule which is a combination of local limits and the prevailing Charging Profiles of the different chargingProfilePurposes and stack levels.	
Purpose	To verify if the CSMS is able to calculate request a composite schedule from the Charging Station.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **GetCompositeScheduleResponse** With **status** *Accepted* **schedule.evseId** *0* **schedule.duration** is *300* **schedule.chargingRateUnit** *<Specified chargingRateUnit from step 1>* **schedule.chargingSchedulePeriod[0].startPeriod** *0* *Note: Multiply limit by 1000 if chargingRateUnit is W* **schedule.chargingSchedulePeriod[0].limit** *10*	**1.** The CSMS sends a **GetCompositeScheduleRequest**
Tool validations	* Step 1: (Message: **GetCompositeScheduleRequest**) **evseId** *0* **duration** is *<Configured duration>* **chargingRateUnit** *<Configured chargingRateUnit>*	
	Post scenario validations: N/a	

Table 599. Test Case Id: TC_K_48_CSMS

Test case name	**Set / Update External Charging Limit (not on a transaction)**	
Test case Id	TC_K_48_CSMS	
Use case Id(s)	K12	
Requirement(s)	N/a	
System under test	CSMS	
Description	A charging schedule or charging limit can be imposed by an external system on the Charging Station for new transactions or on the grid connection. An External Control System sends a charging limit to a Charging Station. This limit is then sent to the CSMS.	
Purpose	To verify if the CSMS is able to receive the request from a charging station and respond correctly as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The OCTT sends a **NotifyChargingLimitRequest** with - **chargingLimit.chargingLimitSource** *EMS*	**2.** The CSMS responds with a **NotifyChargingLimitResponse**
Tool validations	- N/a	
	Post scenario validations: - N/a	

Table 600. Test Case Id: TC_K_50_CSMS

Test case name	**Reset / release external charging limit - Without ongoing transaction**	
Test case Id	TC_K_50_CSMS	
Use case Id(s)	K13	
Requirement(s)	N/a	
System under test	CSMS	
Description	A charging schedule or charging limit can be removed by an external system on the Charging Station. An external control system sends a signal to release a previously imposed charging limit to a Charging Station. The Charging Station notifies the CSMS about this.	
Purpose	To verify if the CSMS is able to receive the notify from a charging station and respond correctly as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The OCTT sends a **ClearedChargingLimitRequest** with - **chargingLimitSource** *EMS*	**2.** The CSMS responds with a **ClearedChargingLimitResponse**
Tool validations	- N/a	
	Post scenario validations: - N/a	

Table 601. Test Case Id: TC_K_51_CSMS

Test case name	**Reset / release external charging limit - With ongoing transaction**	
Test case Id	TC_K_51_CSMS	
Use case Id(s)	K13	
Requirement(s)	N/a	
System under test	CSMS	
Description	A charging schedule or charging limit can be removed by an external system on the Charging Station. An external control system sends a signal to release a previously imposed charging limit to a Charging Station. The Charging Station notifies the CSMS about this.	
Purpose	To verify if the CSMS is able to receive the notify from a charging station and respond correctly as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): State is *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The OCTT sends a **ClearedChargingLimitRequest** with - **chargingLimitSource** *EMS*	**2.** The CSMS responds with a **ClearedChargingLimitResponse**
	3. The OCTT sends a **TransactionEventRequest** with - **eventType** *Updated* - **triggerReason** *ChargingRateChanged*	**4.** The CSMS responds with a **TransactionEventResponse**
Tool validations	- N/a	
	Post scenario validations: - N/a	

Table 602. Test Case Id: TC_K_52_CSMS

Test case name	**Set / Update External Charging Limit (not on a transaction) - ChargingStationExternalConstraints in report**	
Test case Id	TC_K_52_CSMS	
Use case Id(s)	K12	
Requirement(s)	N/a	
System under test	CSMS	
Description	A charging schedule or charging limit can be removed by an external system on the Charging Station. An external control system sends a signal to release a previously imposed charging limit to a Charging Station. The Charging Station notifies the CSMS about this.	
Purpose	To verify if the CSMS is able to correctly receive the report when a charging limit has been externally changed in a charging station as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **GetChargingProfilesResponse** with - **status** *Accepted*	**1.** The CSMS sends a **GetChargingProfilesRequest**
	3. The OCTT sends a **ReportChargingProfilesRequest** with - **requestId** *Generated Id* - **chargingProfile.chargingProfilePurpose** *ChargingStationExternalConstraints*	**4.** The CSMS responds with a **ReportChargingProfilesResponse**
Tool validations	N/a	
	Post scenario validations: - N/a	

Table 603. Test Case Id: TC_K_53_CSMS

Test case name	**Charging with load leveling based on High Level Communication - Success**	
Test case Id	TC_K_53_CSMS	
Use case Id(s)	K15	
Requirement(s)	K15.FR.02,K15.FR.03,K15.FR.05,K15.FR.07,K15.FR.11	
System under test	CSMS	
Description	ISO15118-1 E1 AC Charging with load leveling based on High Level Communication, and E4 DC charging with load leveling based on High Level Communication.	
Purpose	To verify if the CSMS is able to perform load leveling when it receives the EV charging needs from the Charging Station.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): State is *Authorized* State is *EVConnectedPreSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
	Execute reusable state *ISO15118SmartCharging*	
Tool validations	- N/a	
	Post scenario validations: N/a	

Table 604. Test Case Id: TC_K_55_CSMS

Test case name	**Charging with load leveling based on High Level Communication - EV charging profile exceeds limits**	
Test case Id	TC_K_55_CSMS	
Use case Id(s)	K15,K16,K17	
Requirement(s)	K15.FR.12,K15.FR.13,K16.FR.07,K16.FR.08,K17.FR.12,K17.FR.13	
System under test	CSMS	
Description	ISO15118-1 E1 AC Charging with load leveling based on High Level Communication, and E4 DC charging with load leveling based on High Level Communication.	
Purpose	To verify if the CSMS is able to renegotiate when it receives the EV charging schedule which exceeds the profile limits.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): State is *Authorized* State is *EVConnectedPreSession*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The OCTT sends a **NotifyEVChargingNeedsRequest**. With **evseId** *<Configured evseId>* **maxScheduleTuples** & **chargingNeeds** *<Configured values from mock EV>*	**2.** The CSMS responds with a **NotifyEVChargingNeedsResponse**.
	4. The OCTT responds with a **SetChargingProfileResponse** With **status** *Accepted*	**3.** The CSMS sends a **SetChargingProfileRequest**
	5. The OCTT sends a **NotifyEVChargingScheduleRequest**. With **evseId** *<Configured evseId>* **chargingSchedule** *<ChargingSchedule that exceeds the limits of the chargingSchedule provided at step 3.>*	**6.** The CSMS responds with a **NotifyEVChargingScheduleResponse**.
	7. The OCTT sends a **TransactionEventRequest**. With **triggerReason** *ChargingStateChanged* **transactionInfo.chargingState** *Charging*	**8.** The CSMS responds with a **TransactionEventResponse**.
	10. The OCTT responds with a **SetChargingProfileResponse** With **status** *Accepted*	**9.** The CSMS sends a **SetChargingProfileRequest**
	11. The OCTT sends a **NotifyEVChargingScheduleRequest**. With **evseId** *<Configured evseId>* **chargingSchedule** *<ChargingSchedule provided at step 9>*	**12.** The CSMS responds with a **NotifyEVChargingScheduleResponse**.
	13. The OCTT sends a **TransactionEventRequest**. With **triggerReason** *ChargingRateChanged*	**14.** The CSMS responds with a **TransactionEventResponse**.

Test case name	**Charging with load leveling based on High Level Communication - EV charging profile exceeds limits**
Tool validations	* Step 2: (Message: **NotifyEVChargingNeedsResponse**) **status** *Accepted* * Step 3: (Message: **SetChargingProfileRequest**) **evseId** *<Configured evseId>* **chargingProfilePurpose** *TxProfile* **transactionId** *<Provided transactionId from before>* * Step 6: (Message: **NotifyEVChargingScheduleResponse**) **status** *Rejected* * Step 9: (Message: **SetChargingProfileRequest**) **evseId** *<Configured evseId>* **chargingProfilePurpose** *TxProfile* **transactionId** *<Provided transactionId from before>* * Step 12: (Message: **NotifyEVChargingScheduleResponse**) **status** *Accepted*
	Post scenario validations: N/a

Table 605. Test Case Id: TC_K_57_CSMS

Test case name	**Renegotiating a Charging Schedule - Initiated by EV**	
Test case Id	TC_K_57_CSMS	
Use case Id(s)	K17	
Requirement(s)	K17.FR.02,K17.FR.03,K17.FR.05,K17.FR.07,K17.FR.11	
System under test	CSMS	
Description	The EV signals the Charging Station that it wants to renegotiate and it provides new charging needs, which the Charging Station sends to the CSMS. Based on this and other parameters, the CSMS calculates a new charging schedule and sends it via SetChargingProfileRequest to Charging Station, which communicates it to the EV.	
Purpose	To verify if the CSMS is able to renegotiate when the EV signals the Charging Station that it wants to renegotiate and it provides new charging needs, which the Charging Station sends to the CSMS.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): **State** is *Authorized* AND *EVConnectedPreSession* AND *ISO15118SmartCharging*	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The OCTT sends a **NotifyEVChargingNeedsRequest**. With **evseId** *<Configured evseId>* **maxScheduleTuples** & **chargingNeeds** *<Configured values from mock EV>*	**2.** The CSMS responds with a **NotifyEVChargingNeedsResponse**.
	3. The OCTT sends a **TransactionEventRequest**. With **triggerReason** *ChargingRateChanged*	**4.** The CSMS responds with a **TransactionEventResponse**.
	6. The OCTT responds with a **SetChargingProfileResponse** With **status** *Accepted*	**5.** The CSMS sends a **SetChargingProfileRequest** Note(s): *- If* ***NotifyEVChargingNeedsResponseStatus*** *was Processing, the OCTT will wait 60 seconds for the request*
	7. The OCTT sends a **NotifyEVChargingScheduleRequest**. With **evseId** *<Configured evseId>* **chargingSchedule** *<ChargingSchedule provided at step 5>*	**8.** The CSMS responds with a **NotifyEVChargingScheduleResponse**.
	9. The OCTT sends a **TransactionEventRequest**. With **triggerReason** *ChargingRateChanged>*	**10.** The CSMS responds with a **TransactionEventResponse**.

Test case name	**Renegotiating a Charging Schedule - Initiated by EV**
Tool validations	* Step 2: (Message: **NotifyEVChargingNeedsResponse**) **status** *Accepted* or *Processing* * Step 5: (Message: **SetChargingProfileRequest**) **evseId** *<Configured evseId>* **chargingProfilePurpose** *TxProfile* **transactionId** *<Provided transactionId from before>* * Step 8: (Message: **NotifyEVChargingScheduleResponse**) **status** *Accepted*
	Post scenario validations: N/a

Table 606. Test Case Id: TC_K_58_CSMS

Test case name	**Renegotiating a Charging Schedule - Initiated by CSMS**	
Test case Id	TC_K_58_CSMS	
Use case Id(s)	K16	
Requirement(s)	K16.FR.06	
System under test	CSMS	
Description	The CSMS sends a SetChargingProfileRequest to the Charging Station to influence the power or current drawn by the EV. The CSMS calculates a ChargingSchedule to stay within limits which MAY be imposed by an external system.	
Purpose	To verify if the CSMS is able to renegotiate power/current drawn by the EV.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): **State** is *Authorized* AND *EVConnectedPreSession* AND *ISO15118SmartCharging*	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **SetChargingProfileResponse** With **status** *Accepted*	**1.** The CSMS sends a **SetChargingProfileRequest**
	3. The OCTT sends a **NotifyEVChargingScheduleRequest**. With **evseId** *<Configured evseId>* **chargingSchedule** *<ChargingSchedule provided at step 3>*	**4.** The CSMS responds with a **NotifyEVChargingScheduleResponse**.
	5. The OCTT sends a **TransactionEventRequest**. With **triggerReason** *ChargingRateChanged*	**6.** The CSMS responds with a **TransactionEventResponse**.
Tool validations	* Step 1: (Message: **SetChargingProfileRequest**) **evseId** *<Configured evseId>* **chargingProfilePurpose** *TxProfile* **transactionId** *<Provided transactionId from before>* * Step 4: (Message: **NotifyEVChargingScheduleResponse**) **status** *Accepted*	
	Post scenario validations: N/a	

Table 607. Test Case Id: TC_K_59_CSMS

Test case name	**Renegotiating a Charging Schedule - Initiated by CSMS - Send NotifyEVChargingNeeds**	
Test case Id	TC_K_59_CSMS	
Use case Id(s)	K16	
Requirement(s)	K16.FR.12	
System under test	CSMS	
Description	The CSMS sends a SetChargingProfileRequest to the Charging Station to influence the power or current drawn by the EV. The CSMS calculates a ChargingSchedule to stay within limits which MAY be imposed by an external system.	
Purpose	To verify if the CSMS is able to handle a Charging Stations resending the charging needs of the EV.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): **State** is *Authorized* AND *EVConnectedPreSession* AND *ISO15118SmartCharging*	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **SetChargingProfileResponse** With **status** *Accepted*	**1.** The CSMS sends a **SetChargingProfileRequest**
	3. The OCTT sends a **NotifyEVChargingNeedsRequest**. With **evseId** *<Configured evseId>* **maxScheduleTuples** & **chargingNeeds** *<Configured values from mock EV>*	**4.** The CSMS responds with a **NotifyEVChargingNeedsResponse**.
	6. The OCTT responds with a **SetChargingProfileResponse** With **status** *Accepted*	**5.** The CSMS sends a **SetChargingProfileRequest** Note(s): *- If **NotifyEVChargingNeedsResponseStatus** was Processing, the OCTT will wait 60 seconds for the request*
	7. The OCTT sends a **NotifyEVChargingScheduleRequest**. With **evseId** *<Configured evseId>* **chargingSchedule** *<ChargingSchedule provided at step 3>*	**8.** The CSMS responds with a **NotifyEVChargingScheduleResponse**.
	9. The OCTT sends a **TransactionEventRequest**. With **triggerReason** *ChargingRateChanged*	**10.** The CSMS responds with a **TransactionEventResponse**.

Test case name	**Renegotiating a Charging Schedule - Initiated by CSMS - Send NotifyEVChargingNeeds**
Tool validations	* Step 1: (Message: **SetChargingProfileRequest**) **evseId** *<Configured evseId>* **chargingProfilePurpose** *TxProfile* **transactionId** *<Provided transactionId from before>* * Step 4: (Message: **NotifyEVChargingNeedsResponse**) **status** *Accepted* or *Processing* * Step 5: (Message: **SetChargingProfileRequest**) **evseId** *<Configured evseId>* **chargingProfilePurpose** *TxProfile* **transactionId** *<Provided transactionId from before>* * Step 8: (Message: **NotifyEVChargingScheduleResponse**) **status** *Accepted*
	Post scenario validations: N/a

Table 608. Test Case Id: TC_K_70_CSMS

Test case name	**Set Charging Profile - Multiple Profiles**	
Test case Id	TC_K_70_CSMS	
Use case Id(s)	n/a	
Requirement(s)	n/a	
System under test	CSMS	
Description	To enable the CSMS to influence the charging power or current drawn from a specific EVSE or the entire Charging Station over a period of time. The CSMS sends a SetChargingProfileRequest to the Charging Station to influence the power or current drawn by EVs. The CSMS calculates a ChargingSchedule to stay within certain limits, which MAY be imposed by any external system.	
Purpose	To verify if the CSMS is able to set a charging profile with the same ProfileKind, Purpose, and limit, but with a different stackLevel.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **SetChargingProfileResponse** with **status** *Accepted*	**1.** The CSMS sends a **SetChargingProfileRequest** with **stackLevel** *<Configured stackLevel1>*
	4. The OCTT responds with a **SetChargingProfileResponse** with **status** *Accepted*	**3.** The CSMS sends a **SetChargingProfileRequest** with **stackLevel** *<Configured stackLevel2>*
Tool validations	* Step 3: Message **SetChargingProfileRequest** **chargingProfile.id** *<different id for both chargingProfiles>* **chargingProfile.stackLevel** *<different stackLevel for both chargingProfiles>*	
	Post scenario validations: - N/a	

3.13. L Firmware Management

Table 609. Test Case Id: TC_L_01_CSMS

Test case name	**Secure Firmware Update - Installation successful**
Test case Id	TC_L_01_CSMS
Use case Id(s)	L01
Requirement(s)	L01.FR.01,L01.FR.11,L01.FR.15
System under test	CSMS
Description	The CSMS is able to request the Charging Station to securely download and install a new firmware by sending an UpdateFirmwareRequest with a signingCertificate.
Purpose	To verify if the CSMS is able to request the Charging Station to securely download and install a new firmware.
Prerequisite(s)	N/a
Before (Preparations)	**Configuration State:** N/a
	Memory State: N/a
	Reusable State(s): N/a

Test case name	**Secure Firmware Update - Installation successful**	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **UpdateFirmwareResponse** With **status** *Accepted*	**1.** The CSMS sends a **UpdateFirmwareRequest**
	3. The OCTT sends a **FirmwareStatusNotificationRequest**. With **status** *Downloading*	**4.** The CSMS responds with a **FirmwareStatusNotificationResponse**.
	5. The OCTT sends a **FirmwareStatusNotificationRequest**. With **status** *Downloaded*	**6.** The CSMS responds with a **FirmwareStatusNotificationResponse**.
	7. The OCTT sends a **FirmwareStatusNotificationRequest**. With **status** *SignatureVerified*	**8.** The CSMS responds with a **FirmwareStatusNotificationResponse**.
	9. The OCTT sends a **FirmwareStatusNotificationRequest**. With **status** *Installing*	**10.** The CSMS responds with a **FirmwareStatusNotificationResponse**.
	11. The OCTT sends a **FirmwareStatusNotificationRequest**. With **status** *InstallRebooting*	**12.** The CSMS responds with a **FirmwareStatusNotificationResponse**.
	13. The OCTT sends a **BootNotificationRequest** With **reason** *FirmwareUpdate*	**14.** The CSMS responds with a **BootNotificationResponse**
	15. The OCTT notifies the CSMS about the current state of all connectors. Message: **StatusNotificationRequest** **connectorStatus** *Available* Message: **NotifyEventRequest** **trigger** *Delta* **actualValue** *"Available"* **component.name** *"Connector"* **variable.name** *"AvailabilityState"*	**16.** The CSMS responds accordingly.
	17. The OCTT sends a **FirmwareStatusNotificationRequest**. With **status** *Installed*	**18.** The CSMS responds with a **FirmwareStatusNotificationResponse**.
Tool validations	* Step 1: Message **UpdateFirmwareRequest** - **firmware.signingCertificate** *<Configured signingCertificate>* - **firmware.signature** *<Configured signature>* * Step 14: Message **BootNotificationResponse** - **status** *Accepted*	
	Post scenario validations: N/a	

Table 610. Test Case Id: TC_L_02_CSMS

Test case name	**Secure Firmware Update - InstallScheduled**	
Test case Id	TC_L_02_CSMS	
Use case Id(s)	L01	
Requirement(s)	L01.FR.01,L01.FR.11,L01.FR.15	
System under test	CSMS	
Description	The CSMS is able to request the Charging Station to securely download and install a new firmware by sending an UpdateFirmwareRequest with a signingCertificate.	
Purpose	To verify if the CSMS is able to request the Charging Station to securely download a new firmware and install it	
Prerequisite(s)	The CSMS configuration firmware installDateTime needs to be set to a future dateTime.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **UpdateFirmwareResponse** With **status** *Accepted*	**1.** The CSMS sends a **UpdateFirmwareRequest**
	3. The OCTT sends a **FirmwareStatusNotificationRequest**. With **status** *Downloading*	**4.** The CSMS responds with a **FirmwareStatusNotificationResponse**.
	5. The OCTT sends a **FirmwareStatusNotificationRequest**. With **status** *Downloaded*	**6.** The CSMS responds with a **FirmwareStatusNotificationResponse**.
	7. The OCTT sends a **FirmwareStatusNotificationRequest**. With **status** *SignatureVerified*	**8.** The CSMS responds with a **FirmwareStatusNotificationResponse**.
	9. The OCTT sends a **FirmwareStatusNotificationRequest**. With **status** *InstallScheduled*	**10.** The CSMS responds with a **FirmwareStatusNotificationResponse**.
	11. The OCTT sends a **FirmwareStatusNotificationRequest**. With **status** *Installing* Note(s): *- This step will be executed after the given installDateTime from step 1 has been reached.*	**12.** The CSMS responds with a **FirmwareStatusNotificationResponse**.
	13. The OCTT sends a **FirmwareStatusNotificationRequest**. With **status** *InstallRebooting*	**14.** The CSMS responds with a **FirmwareStatusNotificationResponse**.

Test case name	**Secure Firmware Update - InstallScheduled**	
	15. The OCTT sends a **BootNotificationRequest** With **reason** *FirmwareUpdate*	**16.** The CSMS responds with a **BootNotificationResponse**
	17. The OCTT notifies the CSMS about the current state of all connectors. Message: **StatusNotificationRequest** **connectorStatus** *Available* Message: **NotifyEventRequest** **trigger** *Delta* **actualValue** *"Available"* **component.name** *"Connector"* **variable.name** *"AvailabilityState"*	**18.** The CSMS responds accordingly.
	19. The OCTT sends a **FirmwareStatusNotificationRequest**. With **status** *Installed*	**20.** The CSMS responds with a **FirmwareStatusNotificationResponse**.
Tool validations	* Step 1: Message **UpdateFirmwareRequest** - **firmware.installDateTime** *<A dateTime in the future>* * Step 16: Message **BootNotificationResponse** - **status** *Accepted*	
	Post scenario validations: N/a	

Table 611. Test Case Id: TC_L_03_CSMS

Test case name	**Secure Firmware Update - DownloadScheduled**	
Test case Id	TC_L_03_CSMS	
Use case Id(s)	L01	
Requirement(s)	L01.FR.01,L01.FR.11,L01.FR.15	
System under test	CSMS	
Description	The CSMS is able to request the Charging Station to securely download and install a new firmware by sending an UpdateFirmwareRequest with a signingCertificate.	
Purpose	To verify if the CSMS is able to request the Charging Station to schedule securely downloading a new firmware.	
Prerequisite(s)	The CSMS configuration firmware retrieveDateTime needs to be set to a future dateTime.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **UpdateFirmwareResponse** With **status** *Accepted*	**1.** The CSMS sends a **UpdateFirmwareRequest**
	3. The OCTT sends a **FirmwareStatusNotificationRequest**. With **status** *DownloadScheduled*	**4.** The CSMS responds with a **FirmwareStatusNotificationResponse**.
	5. The OCTT sends a **FirmwareStatusNotificationRequest**. With **status** *Downloading* Note(s): *- This step will be executed after the given retrieveDateTime from step 1 has been reached.*	**6.** The CSMS responds with a **FirmwareStatusNotificationResponse**.
	7. The OCTT sends a **FirmwareStatusNotificationRequest**. With **status** *Downloaded*	**8.** The CSMS responds with a **FirmwareStatusNotificationResponse**.
	9. The OCTT sends a **FirmwareStatusNotificationRequest**. With **status** *SignatureVerified*	**10.** The CSMS responds with a **FirmwareStatusNotificationResponse**.
	11. The OCTT sends a **FirmwareStatusNotificationRequest**. With **status** *Installing*	**12.** The CSMS responds with a **FirmwareStatusNotificationResponse**.
	13. The OCTT sends a **FirmwareStatusNotificationRequest**. With **status** *InstallRebooting*	**14.** The CSMS responds with a **FirmwareStatusNotificationResponse**.

Test case name	**Secure Firmware Update - DownloadScheduled**	
	15. The OCTT sends a **BootNotificationRequest** With **reason** *FirmwareUpdate*	**16.** The CSMS responds with a **BootNotificationResponse**
	17. The OCTT notifies the CSMS about the current state of all connectors. Message: **StatusNotificationRequest** **connectorStatus** *Available* Message: **NotifyEventRequest** **trigger** *Delta* **actualValue** *"Available"* **component.name** *"Connector"* **variable.name** *"AvailabilityState"*	**18.** The CSMS responds accordingly.
	19. The OCTT sends a **FirmwareStatusNotificationRequest**. With **status** *Installed*	**20.** The CSMS responds with a **FirmwareStatusNotificationResponse**.
Tool validations	* Step 1: Message **UpdateFirmwareRequest** - **firmware.retrieveDateTime** *<A dateTime in the future>* * Step 16: Message **BootNotificationResponse** - **status** *Accepted*	
	Post scenario validations: N/a	

Table 612. Test Case Id: TC_L_04_CSMS

Test case name	**Secure Firmware Update - RevokedCertificate**	
Test case Id	TC_L_04_CSMS	
Use case Id(s)	L01	
Requirement(s)	L01.FR.01	
System under test	CSMS	
Description	The CSMS is able to request the Charging Station to securely download and install a new firmware by sending an UpdateFirmwareRequest with a signingCertificate.	
Purpose	To verify if the CSMS is able to handle a Charging Station reporting the firmware signing certificate is revoked.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **UpdateFirmwareResponse** With **status** *RevokedCertificate*	**1.** The CSMS sends a **UpdateFirmwareRequest**
Tool validations	N/a	
	Post scenario validations: N/a	

Table 613. Test Case Id: TC_L_05_CSMS

Test case name	**Secure Firmware Update - InvalidCertificate**	
Test case Id	TC_L_05_CSMS	
Use case Id(s)	L01	
Requirement(s)	L01.FR.01	
System under test	CSMS	
Description	The CSMS is able to request the Charging Station to securely download and install a new firmware by sending an UpdateFirmwareRequest with a signingCertificate.	
Purpose	To verify if the CSMS is able to handle a Charging Station reporting the firmware signing certificate is invalid.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **UpdateFirmwareResponse** With **status** *InvalidCertificate*	**1.** The CSMS sends a **UpdateFirmwareRequest**
	3. The OCTT sends a **SecurityEventNotificationRequest** With **type** is *InvalidFirmwareSigningCertificate*	**4.** The CSMS responds with a **SecurityEventNotificationResponse**
Tool validations	N/a	
	Post scenario validations: N/a	

Table 614. Test Case Id: TC_L_06_CSMS

Test case name	**Secure Firmware Update - InvalidSignature**	
Test case Id	TC_L_06_CSMS	
Use case Id(s)	L01	
Requirement(s)	L01.FR.01,L01.FR.11	
System under test	CSMS	
Description	The CSMS is able to request the Charging Station to securely download and install a new firmware by sending an UpdateFirmwareRequest with a signingCertificate.	
Purpose	To verify if the CSMS is able to handle a Charging Station reporting the signature is invalid.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **UpdateFirmwareResponse** With **status** *Accepted*	**1.** The CSMS sends a **UpdateFirmwareRequest**
	3. The OCTT sends a **FirmwareStatusNotificationRequest**. With **status** *Downloading*	**4.** The CSMS responds with a **FirmwareStatusNotificationResponse**.
	5. The OCTT sends a **FirmwareStatusNotificationRequest**. With **status** *Downloaded*	**6.** The CSMS responds with a **FirmwareStatusNotificationResponse**.
	7. The OCTT sends a **FirmwareStatusNotificationRequest**. With **status** *InvalidSignature*	**8.** The CSMS responds with a **FirmwareStatusNotificationResponse**.
	9. The OCTT sends a **SecurityEventNotificationRequest** With **type** is *InvalidFirmwareSignature*	**10.** The CSMS responds with a **SecurityEventNotificationResponse**
Tool validations	N/a	
	Post scenario validations: N/a	

Table 615. Test Case Id: TC_L_07_CSMS

Test case name	**Secure Firmware Update - DownloadFailed**	
Test case Id	TC_L_07_CSMS	
Use case Id(s)	L01	
Requirement(s)	L01.FR.01,L01.FR.11	
System under test	CSMS	
Description	The CSMS is able to request the Charging Station to securely download and install a new firmware by sending an UpdateFirmwareRequest with a signingCertificate.	
Purpose	To verify if the CSMS is able to handle a Charging Station reporting it failed to download the firmware.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **UpdateFirmwareResponse** With **status** *Accepted*	**1.** The CSMS sends a **UpdateFirmwareRequest**
	3. The OCTT sends a **FirmwareStatusNotificationRequest**. With **status** *Downloading*	**4.** The CSMS responds with a **FirmwareStatusNotificationResponse**.
	5. The OCTT sends a **FirmwareStatusNotificationRequest**. With **status** *DownloadFailed*	**6.** The CSMS responds with a **FirmwareStatusNotificationResponse**.
Tool validations	N/a	
	Post scenario validations: N/a	

Table 616. Test Case Id: TC_L_08_CSMS

Test case name	**Secure Firmware Update - InstallVerificationFailed**	
Test case Id	TC_L_08_CSMS	
Use case Id(s)	L01	
Requirement(s)	L01.FR.01,L01.FR.11	
System under test	CSMS	
Description	The CSMS is able to request the Charging Station to securely download and install a new firmware by sending an UpdateFirmwareRequest with a signingCertificate.	
Purpose	To verify if the CSMS is able to handle a Charging Station reporting the verification of the firmware failed during installation.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **UpdateFirmwareResponse** With **status** *Accepted*	**1.** The CSMS sends a **UpdateFirmwareRequest**
	3. The OCTT sends a **FirmwareStatusNotificationRequest**. With **status** *Downloading*	**4.** The CSMS responds with a **FirmwareStatusNotificationResponse**.
	5. The OCTT sends a **FirmwareStatusNotificationRequest**. With **status** *Downloaded*	**6.** The CSMS responds with a **FirmwareStatusNotificationResponse**.
	7. The OCTT sends a **FirmwareStatusNotificationRequest**. With **status** *SignatureVerified*	**8.** The CSMS responds with a **FirmwareStatusNotificationResponse**.
	9. The OCTT sends a **FirmwareStatusNotificationRequest**. With **status** *Installing*	**10.** The CSMS responds with a **FirmwareStatusNotificationResponse**.
	11. The OCTT sends a **FirmwareStatusNotificationRequest**. With **status** *InstallVerificationFailed*	**12.** The CSMS responds with a **FirmwareStatusNotificationResponse**.
Tool validations	N/a	
	Post scenario validations: N/a	

Table 617. Test Case Id: TC_L_09_CSMS

Test case name	**Secure Firmware Update - InstallationFailed**	
Test case Id	TC_L_09_CSMS	
Use case Id(s)	L01	
Requirement(s)	L01.FR.01,L01.FR.11	
System under test	CSMS	
Description	The CSMS is able to request the Charging Station to securely download and install a new firmware by sending an UpdateFirmwareRequest with a signingCertificate.	
Purpose	To verify if the CSMS is able to handle a Charging Station reporting the installation of the firmware failed.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **UpdateFirmwareResponse** With **status** *Accepted*	**1.** The CSMS sends a **UpdateFirmwareRequest**
	3. The OCTT sends a **FirmwareStatusNotificationRequest**. With **status** *Downloading*	**4.** The CSMS responds with a **FirmwareStatusNotificationResponse**.
	5. The OCTT sends a **FirmwareStatusNotificationRequest**. With **status** *Downloaded*	**6.** The CSMS responds with a **FirmwareStatusNotificationResponse**.
	7. The OCTT sends a **FirmwareStatusNotificationRequest**. With **status** *SignatureVerified*	**8.** The CSMS responds with a **FirmwareStatusNotificationResponse**.
	9. The OCTT sends a **FirmwareStatusNotificationRequest**. With **status** *Installing*	**10.** The CSMS responds with a **FirmwareStatusNotificationResponse**.
	11. The OCTT sends a **FirmwareStatusNotificationRequest**. With **status** *InstallRebooting*	**12.** The CSMS responds with a **FirmwareStatusNotificationResponse**.
	13. The OCTT sends a **BootNotificationRequest** With **reason** *FirmwareUpdate*	**14.** The CSMS responds with a **BootNotificationResponse**
	15. The OCTT notifies the CSMS about the current state of all connectors. Message: **StatusNotificationRequest** **connectorStatus** *Available* Message: **NotifyEventRequest** **trigger** *Delta* **actualValue** *"Available"* **component.name** *"Connector"* **variable.name** *"AvailabilityState"*	**16.** The CSMS responds accordingly.
	17. The OCTT sends a **FirmwareStatusNotificationRequest**. With **status** *InstallationFailed*	**18.** The CSMS responds with a **FirmwareStatusNotificationResponse**.

Test case name	**Secure Firmware Update - InstallationFailed**
Tool validations	* Step 14: Message **BootNotificationResponse** - **status** *Accepted*
	Post scenario validations: N/a

Table 618. Test Case Id: TC_L_10_CSMS

Test case name	**Secure Firmware Update - AcceptedCanceled**	
Test case Id	TC_L_10_CSMS	
Use case Id(s)	L01	
Requirement(s)	L01.FR.01,L01.FR.11,L01.FR.24	
System under test	CSMS	
Description	The CSMS is able to request the Charging Station to securely download and install a new firmware by sending an UpdateFirmwareRequest with a signingCertificate.	
Purpose	To verify if the CSMS is able to handle a Charging Station reporting an ongoing installation of a firmware was canceled and it is now starting the new firmware update.	
Prerequisite(s)	The CSMS is able to request a new firmware update, while there is already one ongoing on the Charging Station.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **UpdateFirmwareResponse** With **status** *Accepted*	**1.** The CSMS sends a **UpdateFirmwareRequest**
	3. The OCTT sends a **FirmwareStatusNotificationRequest**. With **status** *Downloading*	**4.** The CSMS responds with a **FirmwareStatusNotificationResponse**.
	6. The OCTT responds with a **UpdateFirmwareResponse** With **status** *AcceptedCanceled*	**5.** The CSMS sends a **UpdateFirmwareRequest**
	7. The OCTT sends a **FirmwareStatusNotificationRequest**. With **status** *Downloading*	**8.** The CSMS responds with a **FirmwareStatusNotificationResponse**.
	9. The OCTT sends a **FirmwareStatusNotificationRequest**. With **status** *Downloaded*	**10.** The CSMS responds with a **FirmwareStatusNotificationResponse**.
	11. The OCTT sends a **FirmwareStatusNotificationRequest**. With **status** *SignatureVerified*	**12.** The CSMS responds with a **FirmwareStatusNotificationResponse**.
	13. The OCTT sends a **FirmwareStatusNotificationRequest**. With **status** *Installing*	**14.** The CSMS responds with a **FirmwareStatusNotificationResponse**.
	15. The OCTT sends a **FirmwareStatusNotificationRequest**. With **status** *InstallRebooting*	**16.** The CSMS responds with a **FirmwareStatusNotificationResponse**.

Test case name	**Secure Firmware Update - AcceptedCanceled**	
	17. The OCTT sends a **BootNotificationRequest** With **reason** *FirmwareUpdate*	**18.** The CSMS responds with a **BootNotificationResponse**
	19. The OCTT notifies the CSMS about the current state of all connectors. Message: **StatusNotificationRequest** **connectorStatus** *Available* Message: **NotifyEventRequest** **trigger** *Delta* **actualValue** *"Available"* **component.name** *"Connector"* **variable.name** *"AvailabilityState"*	**20.** The CSMS responds accordingly.
	21. The OCTT sends a **FirmwareStatusNotificationRequest**. With **status** *Installed*	**22.** The CSMS responds with a **FirmwareStatusNotificationResponse**.
Tool validations	* Step 18: Message **BootNotificationResponse** - **status** *Accepted*	
	Post scenario validations: N/a	

Table 619. Test Case Id: TC_L_11_CSMS

Test case name	**Secure Firmware Update - Unable to cancel**	
Test case Id	TC_L_11_CSMS	
Use case Id(s)	L01	
Requirement(s)	L01.FR.01,L01.FR.11,L01.FR.27	
System under test	CSMS	
Description	The CSMS is able to request the Charging Station to securely download and install a new firmware by sending an UpdateFirmwareRequest with a signingCertificate.	
Purpose	To verify if the CSMS is able to handle a Charging Station reporting the ongoing installation of a firmware cannot be canceled.	
Prerequisite(s)	The CSMS is able to request a new firmware update, while there is already one ongoing on the Charging Station.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **UpdateFirmwareResponse** With **status** *Accepted*	**1.** The CSMS sends a **UpdateFirmwareRequest**
	3. The OCTT sends a **FirmwareStatusNotificationRequest**. With **status** *Downloading*	**4.** The CSMS responds with a **FirmwareStatusNotificationResponse**.
	6. The OCTT responds with a **UpdateFirmwareResponse** With **status** *Rejected*	**5.** The CSMS sends a **UpdateFirmwareRequest**
	7. The OCTT sends a **FirmwareStatusNotificationRequest**. With **status** *Downloaded*	**8.** The CSMS responds with a **FirmwareStatusNotificationResponse**.
	9. The OCTT sends a **FirmwareStatusNotificationRequest**. With **status** *SignatureVerified*	**10.** The CSMS responds with a **FirmwareStatusNotificationResponse**.
	11. The OCTT sends a **FirmwareStatusNotificationRequest**. With **status** *Installing*	**12.** The CSMS responds with a **FirmwareStatusNotificationResponse**.
	13. The OCTT sends a **FirmwareStatusNotificationRequest**. With **status** *InstallRebooting*	**14.** The CSMS responds with a **FirmwareStatusNotificationResponse**.
	15. The OCTT sends a **BootNotificationRequest** With **reason** *FirmwareUpdate*	**16.** The CSMS responds with a **BootNotificationResponse**

Test case name	**Secure Firmware Update - Unable to cancel**	
	17. The OCTT notifies the CSMS about the current state of all connectors. Message: **StatusNotificationRequest** **connectorStatus** *Available* Message: **NotifyEventRequest** **trigger** *Delta* **actualValue** *"Available"* **component.name** *"Connector"* **variable.name** *"AvailabilityState"*	**18.** The CSMS responds accordingly.
	19. The OCTT sends a **FirmwareStatusNotificationRequest**. With **status** *Installed*	**20.** The CSMS responds with a **FirmwareStatusNotificationResponse**.
Tool validations	* Step 16: Message **BootNotificationResponse** - **status** *Accepted*	
	Post scenario validations: N/a	

Table 620. Test Case Id: TC_L_13_CSMS

Test case name	**Secure Firmware Update - Unable to download/install firmware with ongoing transaction - AllowNewSessionsPendingFirmwareUpdate is false**	
Test case Id	TC_L_13_CSMS	
Use case Id(s)	L01	
Requirement(s)	L01.FR.01,L01.FR.11	
System under test	CSMS	
Description	The CSMS is able to request the Charging Station to securely download and install a new firmware by sending an UpdateFirmwareRequest with a signingCertificate.	
Purpose	To verify if the CSMS is able to handle a Charging Station setting connectors to Unavailable while preparing a firmware update when there is a transaction ongoing.	
Prerequisite(s)	The CSMS is able to request a new firmware update when there is a transaction ongoing on the Charging Station.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): State is *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **UpdateFirmwareResponse** With **status** *Accepted*	**1.** The CSMS sends a **UpdateFirmwareRequest**
	3. The OCTT sends a **FirmwareStatusNotificationRequest**. With **status** *DownloadScheduled*	**4.** The CSMS responds with a **FirmwareStatusNotificationResponse**.
	5. The OCTT notifies the CSMS about the state change of all connectors that don't have a running transaction. Message: **StatusNotificationRequest** **connectorStatus** *Unavailable* Message: **NotifyEventRequest** **trigger** *Delta* **actualValue** *"Unavailable"* **component.name** *"Connector"* **variable.name** *"AvailabilityState"*	**6.** The CSMS responds accordingly.
	7. Execute **Reusable State** *StopAuthorized* Note(s) *Wait <configured transaction duration> before executing this step*	
	8. Execute **Reusable State** *EVConnectedPostSession*	
	9. Execute **Reusable State** *EVDisconnected*	
	10. The OCTT sends a **FirmwareStatusNotificationRequest**. With **status** *Downloading* Note(s): *- This step will be executed after the given retrieveDateTime from step 1 has been reached.*	**11.** The CSMS responds with a **FirmwareStatusNotificationResponse**.

Test case name	**Secure Firmware Update - Unable to download/install firmware with ongoing transaction - AllowNewSessionsPendingFirmwareUpdate is false**	
	12. The OCTT sends a **FirmwareStatusNotificationRequest**. With **status** *Downloaded*	**13.** The CSMS responds with a **FirmwareStatusNotificationResponse**.
	14. The OCTT sends a **FirmwareStatusNotificationRequest**. With **status** *SignatureVerified*	**15.** The CSMS responds with a **FirmwareStatusNotificationResponse**.
	16. The OCTT sends a **FirmwareStatusNotificationRequest**. With **status** *Installing*	**17.** The CSMS responds with a **FirmwareStatusNotificationResponse**.
	18. The OCTT sends a **FirmwareStatusNotificationRequest**. With **status** *InstallRebooting*	**19.** The CSMS responds with a **FirmwareStatusNotificationResponse**.
	20. The OCTT sends a **BootNotificationRequest** With **reason** *FirmwareUpdate*	**21.** The CSMS responds with a **BootNotificationResponse**
	22. The OCTT notifies the CSMS about the current state of all connectors. Message: **StatusNotificationRequest** **connectorStatus** *Available* Message: **NotifyEventRequest** **trigger** *Delta* **actualValue** *"Available"* **component.name** *"Connector"* **variable.name** *"AvailabilityState"*	**23.** The CSMS responds accordingly.
	24. The OCTT sends a **FirmwareStatusNotificationRequest**. With **status** *Installed*	**25.** The CSMS responds with a **FirmwareStatusNotificationResponse**.
Tool validations	* Step 1: Message **UpdateFirmwareRequest** - **firmware.signingCertificate** *<configured signingCertificate>* * Step 19: Message **BootNotificationResponse** - **status** *Accepted*	
	Post scenario validations: N/a	

Table 621. Test Case Id: TC_L_17_CSMS

Test case name	**Publish Firmware - Published**	
Test case Id	TC_L_17_CSMS	
Use case Id(s)	L03	
Requirement(s)	N/a	
System under test	CSMS	
Description	The Local Controller downloads and publishes a firmware update at the specified URL. This allows the CSMS to send UpdateFirmwareRequests with the URI pointing to the Local Controller, to any Charging Station connected to the Local Controller. This allows the site to save bandwidth and data on the WAN interface.	
Purpose	To verify if the CSMS is able to publish a firmware on the local controller as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **PublishFirmwareResponse** with **status** *Accepted*	**1.** The CSMS sends a **PublishFirmwareRequest**
	3. The OCTT sends a **PublishFirmwareStatusNotificationRequest** with **status** *Downloading*	**4.** The CSMS responds with a **PublishFirmwareStatusNotificationResponse**
	5. The OCTT sends a **PublishFirmwareStatusNotificationRequest** with **status** *Downloaded*	**6.** The CSMS responds with a **PublishFirmwareStatusNotificationResponse**
	7. The OCTT sends a **PublishFirmwareStatusNotificationRequest** with **status** *ChecksumVerified*	**8.** The CSMS responds with a **PublishFirmwareStatusNotificationResponse**
	9. The OCTT sends a **PublishFirmwareStatusNotificationRequest** with **status** *Published* AND **location** *<Configured firmware_location>*	**10.** The CSMS responds with a **PublishFirmwareStatusNotificationResponse**
Tool validations	* Step 1: Message **PublishFirmwareRequest** - **location** *<Configured firmware_location>*	
	Post scenario validations: - N/a	

Table 622. Test Case Id: TC_L_24_CSMS

Test case name	**Publish Firmware - Download failed**	
Test case Id	TC_L_24_CSMS	
Use case Id(s)	L03	
Requirement(s)	N/a	
System under test	CSMS	
Description	The Local Controller downloads and publishes a firmware update at the specified URL. This allows the CSMS to send UpdateFirmwareRequests with the URI pointing to the Local Controller, to any Charging Station connected to the Local Controller. This allows the site to save bandwidth and data on the WAN interface.	
Purpose	To verify if the CSMS is able to publish a firmware on the local controller as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **PublishFirmwareResponse** with **status** *Accepted*	**1.** The CSMS sends a **PublishFirmwareRequest**
	3. The OCTT sends a **PublishFirmwareStatusNotificationRequest** with **status** *Downloading*	**4.** The CSMS responds with a **PublishFirmwareStatusNotificationResponse**
	5. The OCTT sends a **PublishFirmwareStatusNotificationRequest** with **status** *DownloadFailed*	**6.** The CSMS responds with a **PublishFirmwareStatusNotificationResponse**
Tool validations	* Step 1: Message **PublishFirmwareRequest** - **location** *<Configured firmware_location>*	
	Post scenario validations: - N/a	

Table 623. Test Case Id: TC_L_19_CSMS

Test case name	**Publish Firmware - Invalid Checksum**	
Test case Id	TC_L_19_CSMS	
Use case Id(s)	L03	
Requirement(s)	N/a	
System under test	CSMS	
Description	The Local Controller downloads and publishes a firmware update at the specified URL. This allows the CSMS to send UpdateFirmwareRequests with the URI pointing to the Local Controller, to any Charging Station connected to the Local Controller. This allows the site to save bandwidth and data on the WAN interface.	
Purpose	To verify if the CSMS is able to publish a firmware on the local controller as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **PublishFirmwareResponse** with **status** *Accepted*	**1.** The CSMS sends a **PublishFirmwareRequest**
	3. The OCTT sends a **PublishFirmwareStatusNotificationRequest** with **status** *Downloading*	**4.** The CSMS responds with a **PublishFirmwareStatusNotificationResponse**
	5. The OCTT sends a **PublishFirmwareStatusNotificationRequest** with **status** *Downloaded*	**6.** The CSMS responds with a **PublishFirmwareStatusNotificationResponse**
	7. The OCTT sends a **PublishFirmwareStatusNotificationRequest** with **status** *InvalidChecksum*	**8.** The CSMS responds with a **PublishFirmwareStatusNotificationResponse**
Tool validations	* Step 1: Message **PublishFirmwareRequest** - **location** *<Configured firmware_location>*	
	Post scenario validations: - N/a	

Table 624. Test Case Id: TC_L_20_CSMS

Test case name	**Publish Firmware - PublishFailed**	
Test case Id	TC_L_20_CSMS	
Use case Id(s)	L03	
Requirement(s)	N/a	
System under test	CSMS	
Description	The Local Controller downloads and publishes a firmware update at the specified URL. This allows the CSMS to send UpdateFirmwareRequests with the URI pointing to the Local Controller, to any Charging Station connected to the Local Controller. This allows the site to save bandwidth and data on the WAN interface.	
Purpose	To verify if the CSMS is able to publish a firmware on the local controller as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **PublishFirmwareResponse** with **status** *Accepted*	**1.** The CSMS sends a **PublishFirmwareRequest**
	3. The OCTT sends a **PublishFirmwareStatusNotificationRequest** with **status** *Downloading*	**4.** The CSMS responds with a **PublishFirmwareStatusNotificationResponse**
	5. The OCTT sends a **PublishFirmwareStatusNotificationRequest** with **status** *Downloaded*	**6.** The CSMS responds with a **PublishFirmwareStatusNotificationResponse**
	7. The OCTT sends a **PublishFirmwareStatusNotificationRequest** with **status** *ChecksumVerified*	**8.** The CSMS responds with a **PublishFirmwareStatusNotificationResponse**
	9. The OCTT sends a **PublishFirmwareStatusNotificationRequest** with **status** *PublishFailed*	**10.** The CSMS responds with a **PublishFirmwareStatusNotificationResponse**
Tool validations	* Step 1: Message **PublishFirmwareRequest** - **location** *<Configured firmware_location>*	
	Post scenario validations: - N/a	

Table 625. Test Case Id: TC_L_21_CSMS

Test case name	**Unpublish Firmware - Unpublished**	
Test case Id	TC_L_21_CSMS	
Use case Id(s)	L04	
Requirement(s)	N/a	
System under test	CSMS	
Description	Stop serving a firmware update to connected Charging Stations.	
Purpose	To verify if the CSMS is able to unpublish a firmware on the local controller as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **UnpublishFirmwareResponse** with **status** *Unpublished*	**1.** The CSMS sends a **UnpublishFirmwareRequest**
Tool validations	-N/a	
	Post scenario validations: - N/a	

Table 626. Test Case Id: TC_L_22_CSMS

Test case name	**Unpublish Firmware - NoFirmware**	
Test case Id	TC_L_22_CSMS	
Use case Id(s)	L04	
Requirement(s)	N/a	
System under test	CSMS	
Description	Stop serving a firmware update to connected Charging Stations.	
Purpose	To verify if the CSMS is able to unpublish a firmware on the local controller as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **UnpublishFirmwareResponse** with **status** *NoFirmware*	**1.** The CSMS sends a **UnpublishFirmwareRequest**
Tool validations	-N/a	
	Post scenario validations: - N/a	

Table 627. Test Case Id: TC_L_23_CSMS

Test case name	**Unpublish Firmware - Download Ongoing**	
Test case Id	TC_L_23_CSMS	
Use case Id(s)	L04	
Requirement(s)	N/a	
System under test	CSMS	
Description	Stop serving a firmware update to connected Charging Stations.	
Purpose	To verify if the CSMS is able to unpublish a firmware on the local controller as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **UnpublishFirmwareResponse** with **status** *DownloadOngoing*	**1.** The CSMS sends a **UnpublishFirmwareRequest**
Tool validations	-N/a	
	Post scenario validations: - N/a	

3.14. M ISO 15118 CertificateManagement

Table 628. Test Case Id: TC_M_01_CSMS

Test case name	**Install CA certificate - CSMSRootCertificate**	
Test case Id	TC_M_01_CSMS	
Use case Id(s)	M05	
Requirement(s)	M05.FR.01	
System under test	CSMS	
Description	The CSMS is able to request the Charging Station to install new Root CA certificates using the InstallCertificateRequest message.	
Purpose	To verify if the CSMS is able to request a Charging Station to install a new CSMSRootCertificate.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *CertificateInstalled* for certificateType *CSMSRootCertificate*	
Tool validations	N.a	
	Post scenario validations: N/a	

Table 629. Test Case Id: TC_M_02_CSMS

Test case name	**Install CA certificate - ManufacturerRootCertificate**	
Test case Id	TC_M_02_CSMS	
Use case Id(s)	M05	
Requirement(s)	M05.FR.01	
System under test	CSMS	
Description	The CSMS is able to request the Charging Station to install new Root CA certificates using the InstallCertificateRequest message.	
Purpose	To verify if the CSMS is able to request a Charging Station to install a new ManufacturerRootCertificate.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *CertificateInstalled* for certificateType *ManufacturerRootCertificate*	
Tool validations	N/a	
	Post scenario validations: N/a	

Table 630. Test Case Id: TC_M_03_CSMS

Test case name	**Install CA certificate - V2GRootCertificate**	
Test case Id	TC_M_03_CSMS	
Use case Id(s)	M05	
Requirement(s)	M05.FR.01	
System under test	CSMS	
Description	The CSMS is able to request the Charging Station to install new Root CA certificates using the InstallCertificateRequest message.	
Purpose	To verify if the CSMS is able to request a Charging Station to install a new V2GRootCertificate.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *CertificateInstalled* for certificateType *V2GRootCertificate*	
Tool validations	N/a	
	Post scenario validations: N/a	

Table 631. Test Case Id: TC_M_04_CSMS

Test case name	**Install CA certificate - MORootCertificate**	
Test case Id	TC_M_04_CSMS	
Use case Id(s)	M05	
Requirement(s)	M05.FR.01	
System under test	CSMS	
Description	The CSMS is able to request the Charging Station to install new Root CA certificates using the InstallCertificateRequest message.	
Purpose	To verify if the CSMS is able to request a Charging Station to install a new MORootCertificate.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *CertificateInstalled* for certificateType *MORootCertificate*	
Tool validations	N/a	
	Post scenario validations: N/a	

Table 632. Test Case Id: TC_M_05_CSMS

Test case name	**Install CA certificate - Failed**	
Test case Id	TC_M_05_CSMS	
Use case Id(s)	M05	
Requirement(s)	M05.FR.01,M05.FR.03	
System under test	CSMS	
Description	The CSMS is able to request the Charging Station to install new Root CA certificates using the InstallCertificateRequest message.	
Purpose	To verify if the CSMS is able to handle a Charging Station reporting it failed to install the requested certificate.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Trigger the CSMS to send an InstallCertificateRequest with certificateType CSMSRootCertificate.*	
	2. The OCTT responds with a **InstallCertificateResponse** With **status** is *Failed*	**1.** The CSMS sends a **InstallCertificateRequest**

Table 633. Test Case Id: TC_M_12_CSMS

Test case name	**Retrieve certificates from Charging Station - CSMSRootCertificate**	
Test case Id	TC_M_12_CSMS	
Use case Id(s)	M03	
Requirement(s)	M03.FR.01	
System under test	CSMS	
Description	The CSMS is able to retrieve the certificates installed at the Charging Station using the GetInstalledCertificateIdsRequest message. It supports all available hash algorithms, including SHA256, SHA384, and SHA512.	
Purpose	To verify if the CSMS is able to retrieve the hashData from all CSMSRootCertificates stored at the Charging Station, using all available hash algorithms, including SHA256, SHA384, and SHA512.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *GetInstalledCertificates* for certificateType *CSMSRootCertificate*. The OCTT responds with data hashed with SHA256.	
	2. Execute **Reusable State** *GetInstalledCertificates* for certificateType *CSMSRootCertificate*. The OCTT responds with data hashed with SHA384.	
	3. Execute **Reusable State** *GetInstalledCertificates* for certificateType *CSMSRootCertificate*. The OCTT responds with data hashed with SHA512.	
Tool validations	N/a	
	Post scenario validations: N/a	

Table 634. Test Case Id: TC_M_13_CSMS

Test case name	**Retrieve certificates from Charging Station - ManufacturerRootCertificate**	
Test case Id	TC_M_13_CSMS	
Use case Id(s)	M03	
Requirement(s)	M03.FR.01	
System under test	CSMS	
Description	The CSMS is able to retrieve the certificates installed at the Charging Station using the GetInstalledCertificateIdsRequest message.	
Purpose	To verify if the CSMS is able to retrieve the hashData from all ManufacturerRootCertificate stored at the Charging Station.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *GetInstalledCertificates* for certificateType *ManufacturerRootCertificate*	
Tool validations	N/a	
	Post scenario validations: N/a	

Table 635. Test Case Id: TC_M_14_CSMS

Test case name	**Retrieve certificates from Charging Station - V2GRootCertificate**	
Test case Id	TC_M_14_CSMS	
Use case Id(s)	M03	
Requirement(s)	M03.FR.01	
System under test	CSMS	
Description	The CSMS is able to retrieve the certificates installed at the Charging Station using the GetInstalledCertificateIdsRequest message.	
Purpose	To verify if the CSMS is able to retrieve the hashData from all V2GRootCertificate stored at the Charging Station.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *GetInstalledCertificates* for certificateType *V2GRootCertificate*	
Tool validations	N/a	
	Post scenario validations: N/a	

Table 636. Test Case Id: TC_M_15_CSMS

Test case name	**Retrieve certificates from Charging Station - V2GCertificateChain**	
Test case Id	TC_M_15_CSMS	
Use case Id(s)	M03	
Requirement(s)	M03.FR.01,M03.FR.05	
System under test	CSMS	
Description	The CSMS is able to retrieve the certificates installed at the Charging Station using the GetInstalledCertificateIdsRequest message.	
Purpose	To verify if the CSMS is able to retrieve the hashData from all certificates that are part of a V2GCertificateChain stored at the Charging Station.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *GetInstalledCertificates* for certificateType *V2GCertificateChain*	
Tool validations	N/a	
	Post scenario validations: N/a	

Table 637. Test Case Id: TC_M_16_CSMS

Test case name	**Retrieve certificates from Charging Station - MORootCertificate**	
Test case Id	TC_M_16_CSMS	
Use case Id(s)	M03	
Requirement(s)	M03.FR.01	
System under test	CSMS	
Description	The CSMS is able to retrieve the certificates installed at the Charging Station using the GetInstalledCertificateIdsRequest message.	
Purpose	To verify if the CSMS is able to retrieve the hashData from all MORootCertificate stored at the Charging Station.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *GetInstalledCertificates* for certificateType *MORootCertificate*	
Tool validations	N/a	
	Post scenario validations: N/a	

Table 638. Test Case Id: TC_M_17_CSMS

Test case name	**Retrieve certificates from Charging Station - CSMSRootCertificate & ManufacturerRootCertificate**	
Test case Id	TC_M_17_CSMS	
Use case Id(s)	M03	
Requirement(s)	M03.FR.01	
System under test	CSMS	
Description	The CSMS is able to retrieve the certificates installed at the Charging Station using the GetInstalledCertificateIdsRequest message.	
Purpose	To verify if the CSMS is able to retrieve the hashData from all CSMSRootCertificates and ManufacturerRootCertificate stored at the Charging Station.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. Execute **Reusable State** *GetInstalledCertificates* for certificateType *CSMSRootCertificate* AND *ManufacturerRootCertificate*	
Tool validations	N/a	
	Post scenario validations: N/a	

Table 639. Test Case Id: TC_M_18_CSMS

Test case name	**Retrieve certificates from Charging Station - All certificateTypes**	
Test case Id	TC_M_18_CSMS	
Use case Id(s)	M03	
Requirement(s)	M03.FR.01	
System under test	CSMS	
Description	The CSMS is able to retrieve the certificates installed at the Charging Station using the GetInstalledCertificateIdsRequest message.	
Purpose	To verify if the CSMS is able to retrieve the hashData from all Root CA and V2GCertificateChain certificates stored at the Charging Station.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	<u>Manual Action</u>: *Trigger the CSMS to send a GetInstalledCertificateIdsRequest without certificateType.*	
	2. The OCTT responds with a **GetInstalledCertificateIdsResponse** With **status** is *Accepted* **certificateHashDataChain** contains *<The hashData of all certificates stored at the OCTT truststore>*	**1.** The CSMS sends a **GetInstalledCertificateIdsRequest**
Tool validations	* Step 1: Message: **GetInstalledCertificateIdsRequest** - **certificateType** is omitted	
	Post scenario validations: N/a	

Table 640. Test Case Id: TC_M_19_CSMS

Test case name	**Retrieve certificates from Charging Station - No matching certificate found**	
Test case Id	TC_M_19_CSMS	
Use case Id(s)	M03	
Requirement(s)	M03.FR.01,M03.FR.02	
System under test	CSMS	
Description	The CSMS is able to retrieve the certificates installed at the Charging Station using the GetInstalledCertificateIdsRequest message.	
Purpose	To verify if the CSMS is able to handle a response from the Charging Station indicating it was not able to find a certificate for the requested criteria.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Trigger the CSMS to send a GetInstalledCertificateIdsRequest with certificateType ManufacturerRootCertificate.*	
	2. The OCTT responds with a **GetInstalledCertificateIdsResponse** With **status** is *NotFound* **certificateHashDataChain** is omitted.	**1.** The CSMS sends a **GetInstalledCertificateIdsRequest**
Tool validations	* Step 1: Message: **GetInstalledCertificateIdsRequest** - **certificateType** is *ManufacturerRootCertificate*	
	Post scenario validations: N/a	

Table 641. Test Case Id: TC_M_20_CSMS

Test case name	**Delete a certificate from a Charging Station - Success**	
Test case Id	TC_M_20_CSMS	
Use case Id(s)	M04	
Requirement(s)	M04.FR.01,M04.FR.07	
System under test	CSMS	
Description	The CSMS is able to request the Charging Station to delete an installed certificate using the DeleteCertificateRequest message, using all available hash algorithms, including SHA256, SHA384, and SHA512.	
Purpose	To verify if CSMS is able to request a Charging Station to delete an installed certificate, using all available hash algorithms, including SHA256, SHA384, and SHA512.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. *CertificateInstalled* with certificateType *CSMSRootCertificate*.	
	Manual Action: *Request the CSMS to send a DeleteCertificateRequest.*	
	3. The OCTT responds with a **GetInstalledCertificateIdsResponse** With **status** is *Accepted* **certificateHashDataChain** contains an entry with following values: **certificateHashDataChain[0].certificateType** is *CSMSRootCertificate* **certificateHashDataChain[0].certificateHashData.hashAlgorithm** is *SHA256*	**2.** The CSMS sends a **GetInstalledCertificateIdsRequest**
	5. The OCTT responds with a **DeleteCertificateResponse** With **status** is *Accepted*	**4.** The CSMS sends a **DeleteCertificateRequest**
	Note(s): *- Steps 1 - 5 will be repeated for each hash algorithm (SHA256, SHA384, SHA512).*	
Tool validations	* Step 2: Message: **GetInstalledCertificateIdsRequest** - **certificateType** contains *CSMSRootCertificate* OR is *omitted*. * Step 4: Message: **DeleteCertificateRequest** - **certificateHashData** is *<Returned certificateHashData at Step 3>*.	
	Post scenario validations: N/a	

Table 642. Test Case Id: TC_M_21_CSMS

Test case name	**Delete a certificate from a Charging Station - Failed**	
Test case Id	TC_M_21_CSMS	
Use case Id(s)	M04	
Requirement(s)	M04.FR.01,M04.FR.07	
System under test	CSMS	
Description	The CSMS is able to request the Charging Station to delete an installed certificate using the DeleteCertificateRequest message.	
Purpose	To verify if CSMS is able to handle a Charging Station that fails to delete an installed certificate.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): *CertificateInstalled* with certificateType *CSMSRootCertificate*.	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Request the CSMS to send a DeleteCertificateRequest.*	
	2. The OCTT responds with a **GetInstalledCertificateIdsResponse** With **status** is *Accepted* **certificateHashDataChain** contains an entry with following values: **certificateHashDataChain[0].certificateType** is *CSMSRootCertificate* **certificateHashDataChain[0].certificateHashData.hashAlgorithm** is *SHA256*	**1.** The CSMS sends a **GetInstalledCertificateIdsRequest**
	4. The OCTT responds with a **DeleteCertificateResponse** With **status** is *Failed*	**3.** The CSMS sends a **DeleteCertificateRequest**
Tool validations	* Step 1: Message: **GetInstalledCertificateIdsRequest** - **certificateType** contains *CSMSRootCertificate* OR is *omitted*. * Step 3: Message: **DeleteCertificateRequest** - **certificateHashData** contains *<Returned certificateHashData at Step 2>*.	
	Post scenario validations: N/a	

Table 643. Test Case Id: TC_M_24_CSMS

Test case name	**Get Charging Station Certificate status - Success**	
Test case Id	TC_M_24_CSMS	
Use case Id(s)	M06	
Requirement(s)	M06.FR.01,M06.FR.02,M06.FR.03,M06.FR.08,M06.FR.09	
System under test	CSMS	
Description	The Charging Station is able to request the CSMS to get the status of a (V2G) Charging Station certificate.	
Purpose	To verify if the CSMS is able to provide the status of a requested (V2G) Charging Station certificate.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The OCTT sends one or more subsequent **GetCertificateStatusRequests** With **ocspRequestData** contains *<hashes from configured (V2G) certificate chain SubCA's>*	**2.** The CSMS responds with a **GetCertificateStatusResponse**
Tool validations	Step 2: Message: **GetCertificateStatusResponse** **status** *Accepted* **ocspResult** *<OCSPResponse class as defined in IETF RFC 6960. DER encoded (as defined in IETF RFC 6960), and then base64 encoded.>*	
	Post scenario validations: N/a	

Table 644. Test Case Id: TC_M_26_CSMS

Test case name	**Certificate Installation EV - Success**	
Test case Id	TC_M_26_CSMS	
Use case Id(s)	M01	
Requirement(s)	M01.FR.01	
System under test	CSMS	
Description	The EV initiates installing a new certificate. The Charging Station forwards the request for a new certificate to the CSMS.	
Purpose	To verify if the CSMS is able to return the Raw CertificateInstallationRes response for the EV to the Charging Station.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The OCTT sends a **Get15118EVCertificateRequest** With **action** *Install*	**2.** The CSMS responds with a **Get15118EVCertificateResponse**
Tool validations	* Step 2: Message: **Get15118EVCertificateResponse** - **status** *Accepted* - **exiResponse** *<Raw CertificateInstallationRes response for the EV, Base64 encoded.>*	
	Post scenario validations: N/a	

Table 645. Test Case Id: TC_M_28_CSMS

Test case name	**Certificate Update EV - Success**	
Test case Id	TC_M_28_CSMS	
Use case Id(s)	M02	
Requirement(s)	M02.FR.01	
System under test	CSMS	
Description	The EV initiates updating the existing certificate. The Charging Station forwards the update request to the CSMS.	
Purpose	To verify if the CSMS is able to return the Raw CertificateInstallationRes response for the EV to the Charging Station.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The OCTT sends a **Get15118EVCertificateRequest** With **action** *Update*	**2.** The CSMS responds with a **Get15118EVCertificateResponse**
Tool validations	* Step 2: Message: **Get15118EVCertificateResponse** - **status** *Accepted* - **exiResponse** *<Raw CertificateInstallationRes response for the EV, Base64 encoded.>*	
	Post scenario validations: N/a	

3.15. N Diagnostics

Table 646. Test Case Id: TC_N_01_CSMS

Test case name	**Get Monitoring Report - with monitoringCriteria**	
Test case Id	TC_N_01_CSMS	
Use case Id(s)	N02	
Requirement(s)	N02.FR.05, N02.FR.10	
System under test	CSMS	
Description	CSMS requests a report of monitors that match the component criteria.	
Purpose	To test that CSMS supports requesting a monitoring report for the component criteria and that it handles an empty result set.	
Prerequisite(s)	CS has implemented device model monitoring and MonitoringCtrlr.Enabled = true.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: CSMS requests ClearVariableMonitoring ItemsPerMessage from CS.	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manually instruct CSMS to get a report of monitors for: - all *DeltaMonitoring*	
	2. OCTT responds with: **GetMonitoringReportResponse** with: **Status** *EmptyResultSet*	**1.** CSMS sends **GetMonitoringReportRequest**
	Manually instruct CSMS to get a report of monitors for: - all *ThresholdMonitoring*	
	4. OCTT responds with: **GetMonitoringReportResponse** with: **Status** *Accepted*	**3.** CSMS sends **GetMonitoringReportRequest**
	5. OCTT responds with: **NotifyMonitoringReportRequest**	**6.** CSMS sends **NotifyMonitoringReportResponse**
	Step 5 and 6 are repeated as often as needed to report all configuration variables.	
Tool validations	* Step 1: Message: **GetMonitoringReportRequest** - **monitoringCriteria** = *DeltaMonitoring*	
	* Step 3: Message: **GetMonitoringReportRequest** - **monitoringCriteria** = *ThresholdMonitoring*	
	Post scenario validations: Check that CSMS shows the *Threshold* monitors.	

Table 647. Test Case Id: TC_N_02_CSMS

Test case name	**Get Monitoring Report - with component/variable**	
Test case Id	TC_N_02_CSMS	
Use case Id(s)	N02	
Requirement(s)	N02.FR.05, N02.FR.10	
System under test	CSMS	
Description	CSMS requests a report of monitors that match the the given list of components and variables.	
Purpose	To test that CSMS supports requesting a monitoring report for a given component and variable and that it handles an empty result set.	
Prerequisite(s)	CS has implemented device model monitoring and MonitoringCtrlr.Enabled = true.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: CSMS requests ClearVariableMonitoring ItemsPerMessage from CS.	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manually instruct CSMS to get a report of monitors for: - the variable Power of ChargingStation	
	2. OCTT responds with: **GetMonitoringReportResponse** with: **Status** *EmptyResultSet*	**1.** CSMS sends **GetMonitoringReportRequest**
	Manually instruct CSMS to get a report of monitors for: - the variable AvailabilityState of EVSE #1.	
	4. OCTT responds with: **GetMonitoringReportResponse** with: **Status** *Accepted*	**3.** CSMS sends **GetMonitoringReportRequest**
	5. OCTT responds with: **NotifyMonitoringReportRequest**	**6.** CSMS sends **NotifyMonitoringReportResponse**
	Step 5 and 6 are repeated as often as needed to report all configuration variables.	
Tool validations	* Step 1: Message: **GetMonitoringReportRequest** - **componentVariable[0].component.name** = *"ChargingStation"* - **componentVariable[0].variable.name** = *"Power"*	
	* Step 3: Message: **GetMonitoringReportRequest** - **componentVariable[1].component.name** = *"EVSE"* - **componentVariable[1].component.evse.id** = *1* - **componentVariable[1].variable.name** = *"AvailabilityState"*	
	Post scenario validations: Check that CSMS shows the monitor for AvailabilityState for EVSE #1.	

Table 648. Test Case Id: TC_N_03_CSMS

Test case name	**Get Monitoring Report - with component criteria and component/variable**	
Test case Id	TC_N_03_CSMS	
Use case Id(s)	N02	
Requirement(s)	N02.FR.05, N02.FR.10	
System under test	CSMS	
Description	CSMS requests a report of monitors that match both the component criteria and the given list of components and variables.	
Purpose	To test that CSMS supports requesting a monitoring report for both the component criteria and a given component and variable and that it handles an empty result set.	
Prerequisite(s)	CS has implemented device model monitoring and MonitoringCtrlr.Enabled = true.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: CSMS requests ClearVariableMonitoring ItemsPerMessage from CS.	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manually instruct CSMS to get a report of monitors for: - all *DeltaMonitoring* - and the variable AvailabilityState for EVSE #1.	
	2. OCTT responds with: **GetMonitoringReportResponse** with: **Status** *EmptyResultSet*	**1.** CSMS sends **GetMonitoringReportRequest**
	Manually instruct CSMS to get a report of monitors for: - all *ThresholdMonitoring* - and the variable Power of ChargingStation.	
	4. OCTT responds with: **GetMonitoringReportResponse** with: **Status** *Accepted*	**3.** CSMS sends **GetMonitoringReportRequest**
	5. OCTT responds with: **NotifyMonitoringReportRequest**	**6.** CSMS sends **NotifyMonitoringReportResponse**
	Step 5 and 6 are repeated as often as needed to report all configuration variables.	
Tool validations	* Step 1: Message: **GetMonitoringReportRequest** - **monitoringCriteria** = *DeltaMonitoring* - **componentVariable[0].component.name** = *"EVSE"* - **componentVariable[0].component.evse.id** = *<configured evseId>* - **componentVariable[0].variable.name** = *"AvailabilityState"* * Step 3: Message: **GetMonitoringReportRequest** - **monitoringCriteria** = *ThresholdMonitoring* - **componentVariable[0].component.name** = *"ChargingStation"* - **componentVariable[0].variable.name** = *"Power"*	
	Post scenario validations: Check that CSMS shows the *Threshold* monitors for Power for ChargingStation.	

Table 649. Test Case Id: TC_N_60_CSMS

Test case name	**Get Monitoring Report - with component criteria and list of components/variables**	
Test case Id	TC_N_60_CSMS	
Use case Id(s)	N02	
Requirement(s)	N02.FR.05, N02.FR.10	
System under test	CSMS	
Description	CSMS requests a report of monitors that match both the component criteria and the given list of components and variables.	
Purpose	To test that CSMS supports requesting a monitoring report for both the component criteria and a given list of components and optionally with variables and that it handles an empty result set.	
Prerequisite(s)	CS has implemented device model monitoring and MonitoringCtrlr.Enabled = true.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: CSMS requests ClearVariableMonitoring ItemsPerMessage from CS.	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manually instruct CSMS to get a report of monitors for: - all *ThresholdMonitoring* - and the variable Power of both ChargingStation and EVSE #1.	
	2. OCTT responds with: **GetMonitoringReportResponse** with: **Status** *EmptyResultSet*	**1.** CSMS sends **GetMonitoringReportRequest**
	Manually instruct CSMS to get a report of monitors for: - all *DeltaMonitoring* - and the variable AvailabilityState of both ChargingStation and EVSE #1.	
	4. OCTT responds with: **GetMonitoringReportResponse** with: **Status** *Accepted*	**3.** CSMS sends **GetMonitoringReportRequest**
	5. OCTT responds with: **NotifyMonitoringReportRequest**	**6.** CSMS sends **NotifyMonitoringReportResponse**
	Step 5 and 6 are repeated as often as needed to report all configuration variables.	

Test case name	**Get Monitoring Report - with component criteria and list of components/variables**
Tool validations	* Step 1: Message: **GetMonitoringReportRequest** - **monitoringCriteria** is *DeltaMonitoring* - **componentVariable[0].component.name** = *"ChargingStation"* - **componentVariable[0].variable.name** = *"AvailabilityState"* - **componentVariable[1].component.name** = *"EVSE"* - **componentVariable[1].component.evse.id** = *<configured evseId>* - **componentVariable[1].variable.name** = *"AvailabilityState"* * Step 3: Message: **GetMonitoringReportRequest** - **monitoringCriteria** = *ThresholdMonitoring* - **componentVariable[0].component.name** = *"ChargingStation"* - **componentVariable[0].variable.name** = *"AvailabilityState"* - **componentVariable[1].component.name** = *"EVSE"* - **componentVariable[1].component.evse.id** = *<configured evseId>* - **componentVariable[1].variable.name** = *"AvailabilityState"*
	Post scenario validations: Check that CSMS shows the *Delta* monitors for AvailabilityState for both ChargingStation and EVSE #1.

Table 650. Test Case Id: TC_N_05_CSMS

Test case name	**Set Monitoring Base - success**	
Test case Id	TC_N_05_CSMS	
Use case Id(s)	N03	
Requirement(s)	N03.FR.03, N03.FR.04, N03.FR.05	
System under test	CSMS	
Description	CSMS sends a SetMonitoringBaseRequest for *All*, *FactoryDefault* and *HardWiredOnly*.	
Purpose	To test that CSMS supports all three monitoring base types.	
Prerequisite(s)	CS has implemented device model monitoring and MonitoringCtrlr.Enabled = true.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. OCTT responds with: **SetMonitoringBaseResponse**	*Instruct CSMS to set a monitoring base of _All._* **1.** CSMS sends **SetMonitoringBaseRequest**
	4. OCTT responds with: **SetMonitoringBaseResponse**	*Instruct CSMS to set a monitoring base of _FactoryDefault._* **3.** OCTT sends **SetMonitoringBaseRequest**
	6. The OCTT responds with: **SetMonitoringBaseResponse**	*Instruct CSMS to set a monitoring base of _HardWiredOnly._* **5.** OCTT sends **SetMonitoringBaseRequest**
Tool validations	* Step 1 Message: **SetMonitoringBaseRequest** - **monitoringBase** = *All*	
	* Step 3 Message: **SetMonitoringBaseRequest** - **monitoringBase** = *FactoryDefault*	
	* Step 6 Message: **SetMonitoringBaseRequest** - **monitoringBase** = *HardWiredOnly*	
	Post scenario validations: N/A	

Table 651. Test Case Id: TC_N_08_CSMS

Test case name	**Set Variable Monitoring - One SetMonitoringData element**	
Test case Id	TC_N_08_CSMS	
Use case Id(s)	N04	
Requirement(s)	N04.FR.01, N04.FR.02, N04.FR.17	
System under test	CSMS	
Description	CSMS sends a request to activate monitoring on one variable.	
Purpose	To test that CSMS supports setting monitoring on one variable.	
Prerequisite(s)	CS has implemented device model monitoring and MonitoringCtrlr.Enabled = true.	
Before (Preparations)	**Configuration State:** This test case activates monitoring on the following variable: - Component "EVSE", evse *<Configured evseId>*, variable "AvailabilityState", monitor type *Delta* It assumes, that no monitor is active on this variable prior to the test. *Note 1: this is a required variable for which a monitor can be expected to exist or it can be configured.* *Note 2: Any other component/variable combination that supports monitoring could also be used for this test case.*	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. OCTT responds with: Message: **SetVariableMonitoringResponse** with **setMonitoringResult[0].status** = *Accepted*	**1.** *Request CSMS to install monitors on:* - EVSE #*<Configured evseId>*, AvailabilityState, *Delta*, severity 8
Tool validations	* Step 1: **1.** CSMS sends **SetVariableMonitoringRequest** with: - **setMonitoringData[0].value** = *1*, ← recommended value for *Delta* monitor - **setMonitoringData[0].type** = *Delta*, - **setMonitoringData[0].severity** = *8*, - **setMonitoringData[0].component.name** = *"EVSE"* - **setMonitoringData[0].component.evse.id** = *<Configured evseId>* - **setMonitoringData[0].variable.name** = *"AvailabilityState"*	
	Post scenario validations: N/A	

Table 652. Test Case Id: TC_N_09_CSMS

Test case name	**Set Variable Monitoring - Multiple elements on different component and variable**	
Test case Id	TC_N_09_CSMS	
Use case Id(s)	N04	
Requirement(s)	N04.FR.01, N04.FR.02, N04.FR.17	
System under test	CSMS	
Description	CSMS sends a request to activate monitors on different variables.	
Purpose	To test that CSMS supports setting of multiple monitors on different variables.	
Prerequisite(s)	CS has implemented device model monitoring and MonitoringCtrlr.Enabled = true.	
Before (Preparations)	**Configuration State:** This test case activates monitors on the following variables: - Component "EVSE", evse *<Configured evseId>*, variable "AvailabilityState", monitor type *Delta* - Component "ChargingStation", variable "AvailabilityState", monitor type *Delta* It assumes, that no monitor is active on these variables prior to the test. *Note 1: these are required variables for which a monitor can be expected to exist or it can be configured. Note 2: Any other component/variable combination that supports monitoring could also be used for this test case.*	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. OCTT responds with: Message: **SetVariableMonitoringResponse** with **setMonitoringResult[0].status** = *Accepted*	**1.** *Request CSMS to install monitors on:* - EVSE #*<Configured evseId>*, AvailabilityState, *Delta*, severity 8 - ChargingStation, AvailabilityState, *Delta*, severity 8
Tool validations	* Step 1: **1.** CSMS sends **SetVariableMonitoringRequest** with: - **setMonitoringData[0].value** = *1*, ← recommended value for *Delta* monitor - **setMonitoringData[0].type** = *Delta*, - **setMonitoringData[0].severity** = *8*, - **setMonitoringData[0].component.name** = *"EVSE"* - **setMonitoringData[0].component.evse.id** = *<Configured evseId>* - **setMonitoringData[0].variable.name** = *"AvailabilityState"* - **setMonitoringDate[1].value** = *1*, - **setMonitoringDate[1].type** = *Delta*, - **setMonitoringDate[1].severity** = *8*, - **setMonitoringDate[1].component.name** = *"ChargingStation"* - **setMonitoringDate[1].variable.name** = *"AvailabilityState"*	
	Post scenario validations: N/A	

Table 653. Test Case Id: TC_N_16_CSMS

Test case name	**Set Monitoring Level - Success**	
Test case Id	TC_N_16_CSMS	
Use case Id(s)	N05	
Requirement(s)	N05.FR.01	
System under test	CSMS	
Description	CSMS sets a monitoring level.	
Purpose	To test that CSMS supports setting of a monitoring level.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**2.** OCTT responds with: **SetMonitoringLevelResponse** with **Status** is *Accepted*	**1.** *Instruct CSMS to set a monitoring level with* ***severity*** *= _4*
Tool validations	* Step 1: Message: **SetMonitoringLevelRequest** with: **severity** = *4*	
	Post scenario validations: N/A	

Table 654. Test Case Id: TC_N_17_CSMS

Test case name	**Set Monitoring Level - Out of range**	
Test case Id	TC_N_17_CSMS	
Use case Id(s)	N05	
Requirement(s)	N05.FR.02	
System under test	CSMS	
Description	CSMS sets a monitoring level.	
Purpose	To test that CSMS supports the rejection of setting of a monitoring level.	
Prerequisite(s)	The OCTT will always reject the message, but normally this would only occur if the set severity level is out of range.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**2.** OCTT responds with: **SetMonitoringLevelResponse** with **Status** is *Rejected*	**1.** *Instruct CSMS to set a monitoring level with* ***severity*** *= _4*

Table 655. Test Case Id: TC_N_18_CSMS

Test case name	**Clear Monitoring - Too many elements**	
Test case Id	TC_N_18_CSMS	
Use case Id(s)	N06	
Requirement(s)	N06.FR.04	
System under test	CSMS	
Description	CSMS is requested to clear more monitors than allowed in one request.	
Purpose	To test that CSMS does not exceed the `ItemsPerMessageClearVariableMonitoring` amount of monitors in one request.	
Prerequisite(s)	CS has implemented device model monitoring and MonitoringCtrlr.Enabled = true.	
Before (Preparations)	**Configuration State:** This test requests the value of `ItemsPerMessageClearVariableMonitoring` and then instructs the CSMS to clear (at least) one more monitor than allowed by this value. This value is 'read-only', so it cannot be manipulated in the test. As a consequence, if the Charging Station supports more monitor ids in the list, than can be set by the CSMS, then this cannot be tested.	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**2.** The OCTT responds with: **GetVariablesResponse**	**1.** *Instruct CSMS to send GetVariablesRequest with:* ***Component.name*** *MonitoringCtrlr* ***Variable.name*** *ItemsPerMessage* ***Variable.instance*** *ClearVariableMonitoring.*
	4. The OCTT responds with: **ClearVariableMonitoringResponse**	**3.** *Instruct CSMS to clear more monitors than allowed in ItemsPerMessage.* **ClearVariableMonitoringRequest** with a list of **ids** *Note: these monitor ids do not have to exist.*
Tool validations	* Step 1: Message: Two or more **ClearVariableMonitoringRequest**, so that the maximum number of `ItemsPerMessageClearVariableMonitoring` **ids** is never exceeded. OCTT will reply with a **ClearVariableMonitoringResponse** for each **ClearVariableMonitoringRequest**, but the content of the responses is irrelevant for the test.	
	Post scenario validations: N/A	

Table 656. Test Case Id: TC_N_24_CSMS

Test case name	**Set Variable Monitoring - Periodic event**	
Test case Id	TC_N_24_CSMS	
Use case Id(s)	N08	
Requirement(s)	N08.FR.02	
System under test	CSMS	
Description	Charging Station sends a periodic NotifyEventRequest.	
Purpose	To test that CSMS returns a NotifyEventResponse. *Note: this is identical to TC_N_21_CSMS, only with a periodic event.*	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	Tester makes OCTT send a NotifyEventRequest message.	
	1. OCTT sends **NotifyEventRequest** message.	**2.** CSMS returns **NotifyEventResponse** message.
	Note(s): *- Step 1 and 2 will be repeated n times*	
Tool validations	* Step 2: Message: **NotifyEventResponse** with empty body.	
	Post scenario validations: N/A	

Table 657. Test Case Id: TC_N_25_CSMS

Test case name	**Retrieve Log Information - Diagnostics Log - Success**	
Test case Id	TC_N_25_CSMS	
Use case Id(s)	N01	
Requirement(s)	N/a	
System under test	CSMS	
Description	This test case covers the functionality of getting log information from a Charging Station. The CSMS can request a Charging Station to upload a file with log information to a given location (URL). The format of this log file is not prescribed. The Charging Station successfully uploads a log file and gives information about the status of the upload by sending status notifications to the CSMS.	
Purpose	To verify if the CSMS is able to request a charging station to successfully upload a log as described at the OCPP specification.	
Prerequisite(s)	Charging Station has log information available.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **GetLogResponse** with **status** *Accepted*	**1.** The CSMS sends a **GetLogRequest**
	3. The OCTT sends a **LogStatusNotificationRequest** with - **status** *Uploading* - **requestId** *Same Id as the GetLogRequest*	**4.** The CSMS responds with a **LogStatusNotificationResponse** .
	5. The OCTT sends a **LogStatusNotificationRequest** with - **status** *Uploaded* - **requestId** *Same Id as the GetLogRequest*	**6.** The CSMS responds with a **LogStatusNotificationResponse** .
Tool validations	* Step 1: Message **GetLogRequest** - **logType** *DiagnosticsLog*	
	Post scenario validations: - N/a	

Table 658. Test Case Id: TC_N_27_CSMS

Test case name	**Get Customer Information - Accepted + data**	
Test case Id	TC_N_27_CSMS	
Use case Id(s)	N09	
Requirement(s)	N09.FR.01, N09.FR.04	
System under test	CSMS	
Description	The CSMS sends a message to the Charging Station to retrieve IdToken customer information, for example to be compliant with local privacy laws. The Charging Station notifies the CSMS by sending one or more reports.	
Purpose	To verify if the CSMS sends the request correctly and responds on the notifies as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
		1. The CSMS sends a **CustomerInformationRequest**
	2. The OCTT responds with a **CustomerInformationResponse** with **status** *Accepted*	
	3. The OCTT sends a **NotifyCustomerInformationRequest**	
		4. The CSMS responds with a **NotifyCustomerInformationResponse** .
Tool validations	* Step 1: Message **CustomerInformationRequest** - **report** *true* - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>*	
	Post scenario validations: - N/a	

Table 659. Test Case Id: TC_N_28_CSMS

Test case name	**Get Customer Information - Accepted + no data**	
Test case Id	TC_N_28_CSMS	
Use case Id(s)	N09	
Requirement(s)	N09.FR.01, N09.FR.04	
System under test	CSMS	
Description	The CSMS sends a message to the Charging Station to retrieve IdToken customer information, for example to be compliant with local privacy laws. The Charging Station notifies the CSMS by sending one or more reports.	
Purpose	To verify if the CSMS sends the request correctly and responds on the notifies as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
		1. The CSMS sends a **CustomerInformationRequest**
	2. The OCTT responds with a **CustomerInformationResponse** with **status** *Accepted*	
	3. The OCTT sends a **NotifyCustomerInformationRequest**	
		4. The CSMS responds with a **NotifyCustomerInformationResponse** .
Tool validations	* Step 1: Message **CustomerInformationRequest** - **report** *true* - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>*	
	Post scenario validations: - N/a	

Table 660. Test Case Id: TC_N_29_CSMS

Test case name	**Get Customer Information - Not Accepted**	
Test case Id	TC_N_29_CSMS	
Use case Id(s)	N09	
Requirement(s)	N09.FR.01, N09.FR.04	
System under test	CSMS	
Description	The CSMS sends a message to the Charging Station to retrieve IdToken customer information, but the Charging Station rejects the request.	
Purpose	To verify if the CSMS sends the request correctly as described at the OCPP specification, and can handle the Charging Station rejecting the request.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
		1. The CSMS sends a **CustomerInformationRequest**
	2. The OCTT responds with a **CustomerInformationResponse** with **status** *Rejected*	
Tool validations	* Step 1: Message **CustomerInformationRequest** - **report** *true* - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>*	
	Post scenario validations: - N/a	

Table 661. Test Case Id: TC_N_30_CSMS

Test case name	**Clear Customer Information - Clear and report + data**	
Test case Id	TC_N_30_CSMS	
Use case Id(s)	N10	
Requirement(s)	N10.FR.08	
System under test	CSMS	
Description	The CSMS sends a message to the Charging Station to clear (and retrieve) IdToken customer information, for example to be compliant with local privacy laws. The Charging Station notifies the CSMS by sending one or more reports.	
Purpose	To verify if the CSMS sends the request correctly and responds on the notifies as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
		1. The CSMS sends a **CustomerInformationRequest**
	2. The OCTT responds with a **CustomerInformationResponse** with **status** *Accepted*	
	3. The OCTT sends a **NotifyCustomerInformationRequest**	
		4. The CSMS responds with a **NotifyCustomerInformationResponse** .
Tool validations	* Step 1: Message **CustomerInformationRequest** - **report** *true* - **clear** *true* - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>*	
	Post scenario validations: - N/a	

Table 662. Test Case Id: TC_N_31_CSMS

Test case name	**Clear Customer Information - Clear and report + no data**	
Test case Id	TC_N_31_CSMS	
Use case Id(s)	N10	
Requirement(s)	N10.FR.08	
System under test	CSMS	
Description	The CSMS sends a message to the Charging Station to clear (and retrieve) IdToken customer information, for example to be compliant with local privacy laws. The Charging Station notifies the CSMS by sending one or more reports.	
Purpose	To verify if the CSMS sends the request correctly and responds on the notifies as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **CustomerInformationResponse** with **status** *Accepted*	**1.** The CSMS sends a **CustomerInformationRequest**
	3. The OCTT sends a **NotifyCustomerInformationRequest**	**4.** The CSMS responds with a **NotifyCustomerInformationResponse** .
Tool validations	* Step 1: Message **CustomerInformationRequest** - **report** *true* - **clear** *true* - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>*	
	Post scenario validations: - N/a	

Table 663. Test Case Id: TC_N_32_CSMS

Test case name	**Clear Customer Information - Clear and no report**	
Test case Id	TC_N_32_CSMS	
Use case Id(s)	N10	
Requirement(s)	N10.FR.08	
System under test	CSMS	
Description	The CSMS sends a message to the Charging Station to clear IdToken customer information, for example to be compliant with local privacy laws.	
Purpose	To verify if the CSMS sends the request correctly.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
		1. The CSMS sends a **CustomerInformationRequest**
	2. The OCTT responds with a **CustomerInformationResponse** with **status** *Accepted*	
	3. The OCTT sends a **NotifyCustomerInformationRequest**	
		4. The CSMS responds with a **NotifyCustomerInformationResponse**
Tool validations	* Step 1: Message **CustomerInformationRequest** - **report** *false* - **clear** *true* - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>*	
	Post scenario validations: - N/a	

Table 664. Test Case Id: TC_N_62_CSMS

Test case name	**Clear Customer Information - Clear and report - customerIdentifier**	
Test case Id	TC_N_62_CSMS	
Use case Id(s)	N10	
Requirement(s)	N10.FR.08	
System under test	CSMS	
Description	The CSMS sends a message to the Charging Station to clear (and retrieve) raw customer information, for example to be compliant with local privacy laws. The Charging Station notifies the CSMS by sending one or more reports.	
Purpose	To verify if the CSMS sends the request correctly and responds on the notifies as described at the OCPP specification.	
Prerequisite(s)	The CSMS supports retrieving / deleting CustomerInformation - CustomerIdentifier	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
		1. The CSMS sends a **CustomerInformationRequest**
	2. The OCTT responds with a **CustomerInformationResponse** with **status** *Accepted*	
	3. The OCTT sends a **NotifyCustomerInformationRequest**	
		4. The CSMS responds with a **NotifyCustomerInformationResponse**
Tool validations	* Step 1: Message **CustomerInformationRequest** - **report** *true* - **clear** *true* - **customerIdentifier** *"OpenChargeAlliance"*	
	Post scenario validations: - N/a	

Table 665. Test Case Id: TC_N_63_CSMS

Test case name	**Clear Customer Information - Clear and report - customerCertificate**	
Test case Id	TC_N_63_CSMS	
Use case Id(s)	N10	
Requirement(s)	N10.FR.08	
System under test	CSMS	
Description	The CSMS sends a message to the Charging Station to clear (and retrieve) a customer certificate, for example to be compliant with local privacy laws. The Charging Station notifies the CSMS by sending one or more reports.	
Purpose	To verify if the CSMS sends the request correctly and responds on the notifies as described at the OCPP specification.	
Prerequisite(s)	The CSMS supports retrieving / deleting CustomerInformation - CustomerCertificate	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **CustomerInformationResponse** with **status** *Accepted*	**1.** The CSMS sends a **CustomerInformationRequest** with specific hash data *<customer certificate hash data>*.
	3. The OCTT sends a **NotifyCustomerInformationRequest**	**4.** The CSMS responds with a **NotifyCustomerInformationResponse**
Tool validations	* Step 1: Message **CustomerInformationRequest** - **report** *true* - **clear** *true* - **customerCertificate** contains *<customer certificate hash data>*	
	Post scenario validations: - N/a	

Table 666. Test Case Id: TC_N_34_CSMS

Test case name	**Retrieve Log Information - Rejected**	
Test case Id	TC_N_34_CSMS	
Use case Id(s)	N01	
Requirement(s)	N/a	
System under test	CSMS	
Description	This test case covers the functionality of getting log information from a Charging Station. The CSMS can request a Charging Station to upload a file with log information to a given location (URL). The format of this log file is not prescribed. The Charging Station successfully uploads a log file and gives information about the status of the upload by sending status notifications to the CSMS.	
Purpose	To verify if the CSMS is able to request a charging station to successfully upload a log as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **GetLogResponse** with **status** *Rejected*	**1.** The CSMS sends a **GetLogRequest**
Tool validations	N/a	
	Post scenario validations: - N/a	

Table 667. Test Case Id: TC_N_35_CSMS

Test case name	**Retrieve Log Information - Security Log - Success**	
Test case Id	TC_N_35_CSMS	
Use case Id(s)	N01	
Requirement(s)		
System under test	CSMS	
Description	This test case covers the functionality of getting log information from a Charging Station. The CSMS can request a Charging Station to upload a file with log information to a given location (URL). The format of this log file is not prescribed. The Charging Station successfully uploads a log file and gives information about the status of the upload by sending status notifications to the CSMS.	
Purpose	To verify if the CSMS is able to request a charging station to successfully upload a log as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: Charging Station has log information available.	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **GetLogResponse** with **status** *Accepted*	**1.** The CSMS sends a **GetLogRequest**
	3. The OCTT sends a **LogStatusNotificationRequest** with - **status** *Uploading* - **requestId** *Same Id as the GetLogRequest*	**4.** The CSMS responds with a **LogStatusNotificationResponse** .
	5. The OCTT sends a **LogStatusNotificationRequest** with - **status** *Uploaded* - **requestId** *Same Id as the GetLogRequest*	**6.** The OCTT responds with a **LogStatusNotificationResponse** .
Tool validations	* Step 1: Message **GetLogRequest** - **logType** *SecurityLog*	
	Post scenario validations: - N/a	

Table 668. Test Case Id: TC_N_36_CSMS

Test case name	**Retrieve Log Information - Second Request**	
Test case Id	TC_N_36_CSMS	
Use case Id(s)	N01	
Requirement(s)	N/a	
System under test	CSMS	
Description	This test case covers the functionality of getting log information from a Charging Station. The CSMS can request a Charging Station to upload a file with log information to a given location (URL). The format of this log file is not prescribed. The Charging Station successfully uploads a log file and gives information about the status of the upload by sending status notifications to the CSMS.	
Purpose	To verify if the CSMS is able to request a second request while the charging station is uploading a log as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: Charging Station has log information available.	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **GetLogResponse** with **status** *Accepted*	**1.** The CSMS sends a **GetLogRequest**
	3. The OCTT sends a **LogStatusNotificationRequest** with - **status** *Uploading* - **requestId** *Same Id as the GetLogRequest from Step 1*	**4.** The CSMS responds with a **LogStatusNotificationResponse**.
	6. The OCTT responds with a **GetLogResponse** with **status** *AcceptedCanceled*	**5.** The CSMS sends a **GetLogRequest**
	7. The OCTT sends a **LogStatusNotificationRequest** with - **status** *AcceptedCanceled* - **requestId** *Same Id as the GetLogRequest from Step 1*	**8.** The CSMS responds with a **LogStatusNotificationResponse**.
	9. The OCTT sends a **LogStatusNotificationRequest** with - **status** *Uploading* - **requestId** *Same Id as the GetLogRequest from Step 5*	**10.** The CSMS responds with a **LogStatusNotificationResponse** .
	11. The OCTT sends a **LogStatusNotificationRequest** with - **status** *Uploaded* - **requestId** *Same Id as the GetLogRequest from Step 5*	**12.** The CSMS responds with a **LogStatusNotificationResponse** .
Tool validations	N/a	
	Post scenario validations: - N/a	

Table 669. Test Case Id: TC_N_44_CSMS

Test case name	**Clear Monitoring - Rejected**	
Test case Id	TC_N_44_CSMS	
Use case Id(s)	N06	
Requirement(s)	N/a	
System under test	CSMS	
Description	A monitoring setting can be cleared (removed) by sending a ClearVariableMonitoringRequest with the id of the monitoring setting.	
Purpose	To verify if the CSMS is able to correctly read the respond from a charging station on a request to clear a monitor that cannot be cleared as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **ClearVariableMonitoringResponse** with **clearMonitoringResult[0].status** *Rejected*	**1.** The CSMS sends a **ClearVariableMonitoringRequest**
Tool validations	N/a	
	Post scenario validations: - N/a	

Table 670. Test Case Id: TC_N_46_CSMS

Test case name	**Clear Customer Information - Update Local Authorization List**	
Test case Id	TC_N_46_CSMS	
Use case Id(s)	N10	
Requirement(s)	N10.FR.02, N10.FR.08, D01.FR.01, D01.FR.06, D01.FR.18,	
System under test	CSMS	
Description	The CSMS sends a message to the Charging Station to clear (and retrieve) raw customer information, for example to be compliant with local privacy laws. The Charging Station notifies the CSMS by sending one or more reports.	
Purpose	To verify if the CSMS updates the local authorization list when custumor information, which was present in the local authorization list, has been removed as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: A local authorization list with *<Configured customerIdentifier>* is configured.	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
		1. The CSMS sends a **CustomerInformationRequest**
	2. The OCTT responds with a **CustomerInformationResponse** with **status** *Accepted*	
	3. The OCTT sends a **NotifyCustomerInformationRequest**	
		4. The CSMS responds with a **NotifyCustomerInformationResponse** .
		5. The CSMS sends a **SendLocalListRequest**
	6 The OCTT responds with a **SendLocalListResponse** with **status** *Accepted*	
	Note(s): *If the Local Authorization List is too big for one message, step 5 and 6 will be repeated*	
Tool validations	* Step 1: Message **CustomerInformationRequest** - **report** *true* AND - **clear** *true* AND - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>* * Step 5: Message **SendLocalListRequest** - **updateType** *Differential* - **versionNumber** *<Bigger than currently configured in OCTT>* - **localAuthorizationList** *<Not empty>*	
	Post scenario validations: - All messages have been received	

Table 671. Test Case Id: TC_N_47_CSMS

Test case name	**Get Monitoring report - Report all**	
Test case Id	TC_N_47_CSMS	
Use case Id(s)	N02	
Requirement(s)	N/a	
System under test	CSMS	
Description	This test case describes how the CSMS requests the Charging Station to send a report about configured monitoring settings per component and variable. Optionally, this list can be filtered on monitoringCriteria and componentVariables.	
Purpose	To verify if the CSMS is able to send a get monitor request omitting the monitoringCriteria and componentVariable as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
		1. The CSMS sends a **GetMonitoringReportRequest**
	2. The OCTT responds with a **GetMonitoringReportResponse**	
	3. The OCTT sends a **NotifyMonitoringReportRequest**	
		4. The CSMS responds with a **NotifyMonitoringReportResponse** .
	Note(s): - *If **tbc** is True at Step 3 then step 3 and 4 will be repeated*	
Tool validations	* Step 1: Message **GetMonitoringReportRequest** - **monitoringCriteria** omitted AND - **componentVariable** omitted.	
	Post scenario validations: - N/a	

Table 672. Test Case Id: TC_N_48_CSMS

Test case name	**Alert Event - Variable monitoring on write only**	
Test case Id	TC_N_48_CSMS	
Use case Id(s)	N07	
Requirement(s)	N/a	
System under test	CSMS	
Description	NotifyEventRequest reports every Component/Variable for which a VariableMonitoring setting was triggered. Only the VariableMonitoring settings that are responsible for triggering an event are included.	
Purpose	To verify if the CSMS is able to read a request from a trigger from a variablemonitor which is write only as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The OCTT sends a **NotifyEventRequest** with **eventData.actualValue** empty	**2.** The CSMS responds with a **NotifyEventResponse** .
Tool validations	N/a	
	Post scenario validations: - N/a	

Table 673. Test Case Id: TC_N_49_CSMS

Test case name	**Alert Event - LowerThreshold/UpperThreshold cleared after reboot**	
Test case Id	TC_N_49_CSMS	
Use case Id(s)	N07	
Requirement(s)	N/a	
System under test	CSMS	
Description	NotifyEventRequest reports every Component/Variable for which a VariableMonitoring setting was triggered. Only the VariableMonitoring settings that are responsible for triggering an event are included.	
Purpose	To verify if the CSMS is able to read a request when a trigger is cleared after a reboot as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The OCTT sends a **NotifyEventRequest** with **eventData.cleared** *true*	**2.** The CSMS responds with a **NotifyEventResponse** .
Tool validations	N/a	
	Post scenario validations: - N/a	

Table 674. Test Case Id: TC_N_50_CSMS

Test case name	**Alert Event - Periodic Triggered**	
Test case Id	TC_N_50_CSMS	
Use case Id(s)	N07	
Requirement(s)	N/a	
System under test	CSMS	
Description	NotifyEventRequest reports every Component/Variable for which a VariableMonitoring setting was triggered. Only the VariableMonitoring settings that are responsible for triggering an event are included.	
Purpose	To verify if the CSMS is able to read a request when a trigger reason is periodic after a reboot as described at the OCPP specification.	
Prerequisite(s)	n/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The OCTT sends a **NotifyEventRequest** with **eventData.trigger** *Periodic*	**2.** The CSMS responds with a **NotifyEventResponse** .
Tool validations	N/a	
	Post scenario validations: - N/a	

3.16. O Display Message

Table 675. Test Case Id: TC_O_01_CSMS

Test case name	**Set Display Message - Success**	
Test case Id	TC_O_01_CSMS	
Use case Id(s)	O01	
Requirement(s)	O01_FR_04	
System under test	CSMS	
Description	This test case describes how the CSMS can be requested to sent an SetDisplayMessageRequest to the charging station. Depending on the given parameters the message shall be displayed a certain way and at a certain moment on the Charging Station. These messages are displayed additionally on a Charging Station and are not part of the firmware.	
Purpose	To verify if the CSMS is able to send the request according to the DisplayMessage mechanism as described in the OCPP specification.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Request the CSMS to send a SetDisplayMessageRequest.*	
	2. The OCTT responds with a **SetDisplayMessageResponse** with: **status** *Accepted*	**1.** The CSMS sends a **SetDisplayMessageRequest** with: **state** *<Configured display message state>*
Tool validations	* Step 1: Message **SetDisplayMessageRequest** - **message.id** *<Generated Id>* - **message.priority** *<Configured Priority>* - **message.message.format** *<Configured Format>* - **message.state** *<Configured State>*	
	Post scenario validations: - N/a	

Table 676. Test Case Id: TC_O_02_CSMS

Test case name	**Get all Display Messages - Success**	
Test case Id	TC_O_02_CSMS	
Use case Id(s)	O03	
Requirement(s)	N/a	
System under test	CSMS	
Description	This test case describes how a CSO can request all the installed DisplayMessages configured via OCPP in a Charging Station. The Charging Station can remove messages when they are out-dated, or transactions have ended. It can be very useful for a CSO to be able to view to current list of messages, so the CSO knows which messages are (still) configured.	
Purpose	To verify if the CSMS is able to send the request to get the DisplayMessages according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: A display message is configured.	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **GetDisplayMessagesResponse** with **status** *Accepted*	**1.** The CSMS sends a **GetDisplayMessagesRequest**
	3. The OCTT sends a **NotifyDisplayMessagesRequest**	**4.** The CSMS responds with a **NotifyDisplayMessagesResponse** .
Tool validations	* Step 1: Message **GetDisplayMessagesRequest** - **requestId** *<Generated Id>* - **id** *<Omitted>* - **priority** *<Omitted>* - **state** *<Omitted>*	
	Post scenario validations: - N/a	

Table 677. Test Case Id: TC_O_03_CSMS

Test case name	**Get all Display Messages - No DisplayMessages configured**	
Test case Id	TC_O_03_CSMS	
Use case Id(s)	O03	
Requirement(s)	N/a	
System under test	CSMS	
Description	This test case describes how a CSO can request all the installed DisplayMessages configured via OCPP in a Charging Station. The Charging Station can remove messages when they are out-dated, or transactions have ended. It can be very useful for a CSO to be able to view to current list of messages, so the CSO knows which messages are (still) configured.	
Purpose	To verify if the CSMS can request to get all display messages according to the DisplayMessage mechanism as described in the OCPP specification when no messages are configured.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **GetDisplayMessagesResponse** with **status** *Unknown*	**1.** The CSMS sends a **GetDisplayMessagesRequest**
Tool validations	* Step 1: Message **GetDisplayMessagesRequest** - **requestId** *<Generated request id>*	
	Post scenario validations: - N/a	

Table 678. Test Case Id: TC_O_04_CSMS

Test case name	**Clear Display Message - Success**	
Test case Id	TC_O_04_CSMS	
Use case Id(s)	O05	
Requirement(s)	N/a	
System under test	CSMS	
Description	This test case describes how a CSO can remove a specific message, configured via OCPP in a Charging Station.	
Purpose	To verify if the CSMS is able to request the Charging Station to clear a message according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: A display message is configured.	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	Note: *As a help method, a GetDisplayMessagesRequest is requested first for CSMS's that implemented their ClearDisplayMessage as a combined feature.*	
	2. The OCTT responds with a **ClearDisplayMessageResponse** with **status** *Accepted*	**1.** The CSMS sends a **ClearDisplayMessageRequest**
Tool validations	* Step 1: Message **ClearDisplayMessageRequest** - **id** *<Generated Id from set display message>*	
	Post scenario validations: - N/a	

Table 679. Test Case Id: TC_O_05_CSMS

Test case name	**Clear Display Message - Unknown Key**	
Test case Id	TC_O_05_CSMS	
Use case Id(s)	O05	
Requirement(s)	N/a	
System under test	CSMS	
Description	This test case describes how a CSO can remove a specific message, configured via OCPP in a Charging Station.	
Purpose	To verify if the CSMS is able to request the Charging Station to clear a message according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	If the CSMS supports sending a ClearDisplayMessageRequest with an unknown id.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **ClearDisplayMessageResponse** with **status** *Unknown*	**1.** The CSMS sends a **ClearDisplayMessageRequest**
Tool validations	N/a	
	Post scenario validations: - N/a	

Table 680. Test Case Id: TC_O_06_CSMS

Test case name	**Set Display Message - Specific transaction - Success**	
Test case Id	TC_O_06_CSMS	
Use case Id(s)	O02	
Requirement(s)	N/a	
System under test	CSMS	
Description	This test case describes how a CSO can set a message to be displayed on a Charging Station for a specific transaction. Depending on the given parameters the message shall be displayed a certain way on the Charging Station.	
Purpose	To verify if the CSMS is able to send a display message correctly according the mechanism as described in the OCPP specification for a specific transaction.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: **State** is *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Request the CSMS to send a display message for a specific transaction.*	
	2. The OCTT responds with a **SetDisplayMessageResponse** with **status** *Accepted*	**1.** The CSMS sends a **SetDisplayMessageRequest**
	3. Execute **Reusable State** *EVDisconnected*	
Tool validations	* Step 1: Message **SetDisplayMessageRequest** - **message.transactionId** *Same ID as previously returned by the Charging Station* AND - **message.priority** *<Configured Priority>*	
	Post scenario validations: - N/a	

Table 681. Test Case Id: TC_O_07_CSMS

Test case name	**Get a Specific Display Message - Id**	
Test case Id	TC_O_07_CSMS	
Use case Id(s)	O04	
Requirement(s)	N/a	
System under test	CSMS	
Description	This test case describes how a CSO can request specific installed DisplayMessages configured via OCPP in a Charging Station. The Charging Station can remove messages when they are out-dated, or transactions have ended. It can be very useful for a CSO to be able to view to current list of messages, so the CSO knows which messages are (still) configured.	
Purpose	To verify if the CSMS is able to request a specific id message from the charging station according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: A display message is configured.	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **GetDisplayMessagesResponse** with **status** *Accepted*	**1.** The CSMS sends a **GetDisplayMessagesRequest**
	3. The OCTT sends a **NotifyDisplayMessagesRequest**	**4.** The CSMS responds with a **NotifyDisplayMessagesResponse** .
Tool validations	* Step 1: Message **GetDisplayMessagesRequest** - **id** *<Configured_Id>* - **priority** *<Omitted>* - **state** *<Omitted>* - **requestId** *<Generated Id>*	
	Post scenario validations: - N/a	

Table 682. Test Case Id: TC_O_08_CSMS

Test case name	**Get a Specific Display Message - Priority**	
Test case Id	TC_O_08_CSMS	
Use case Id(s)	O04	
Requirement(s)	N/a	
System under test	CSMS	
Description	This test case describes how a CSO can request specific installed DisplayMessages configured via OCPP in a Charging Station. The Charging Station can remove messages when they are out-dated, or transactions have ended. It can be very useful for a CSO to be able to view to current list of messages, so the CSO knows which messages are (still) configured.	
Purpose	To verify if the CSMS is able to request specific priority messages from the charging station according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: A message with <Configured_Priority> is configured	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **GetDisplayMessagesResponse** with **status** *Accepted*	**1.** The CSMS sends a **GetDisplayMessagesRequest**
	3. The OCTT sends a **NotifyDisplayMessagesRequest**	**4.** The CSMS responds with a **NotifyDisplayMessagesResponse** .
Tool validations	* Step 1: Message **GetDisplayMessagesRequest** - **priority** *<Configured_Priority>* - **id** *<Omitted>* - **state** *<Omitted>* - **requestId** *<Generated Id>*	
	Post scenario validations: - N/a	

Table 683. Test Case Id: TC_O_09_CSMS

Test case name	**Get a Specific Display Message - State**	
Test case Id	TC_O_09_CSMS	
Use case Id(s)	O04	
Requirement(s)	N/a	
System under test	CSMS	
Description	This test case describes how a CSO can request specific installed DisplayMessages configured via OCPP in a Charging Station. The Charging Station can remove messages when they are out-dated, or transactions have ended. It can be very useful for a CSO to be able to view to current list of messages, so the CSO knows which messages are (still) configured.	
Purpose	To verify if the CSMS is able to request specific state messages from the charging station according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: A message with <Configured_State> is configured	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **GetDisplayMessagesResponse** with **status** *Accepted*	**1.** The CSMS sends a **GetDisplayMessagesRequest**
	3. The OCTT sends a **NotifyDisplayMessagesRequest**	**4.** The CSMS responds with a **NotifyDisplayMessagesResponse** .
Tool validations	* Step 1: Message **GetDisplayMessagesRequest** - **state** *<Configured_State>* - **priority** *<Omitted>* - **id** *<Omitted>* - **requestId** *<Generated Id>*	
	Post scenario validations: - N/a	

Table 684. Test Case Id: TC_O_10_CSMS

Test case name	**Set Display Message - Specific transaction - UnknownTransaction**	
Test case Id	TC_O_10_CSMS	
Use case Id(s)	O02	
Requirement(s)	N/a	
System under test	CSMS	
Description	This test case describes how a CSMS can attempt to set a DisplayMessage for a transactionId that the CS does not know. The CS will respond with a SetDisplayMessageResponse status of UnknownTransaction.	
Purpose	To verify if the CSMS is able to send a display message correctly according the mechanism as described in the OCPP specification for a specific transaction.	
Prerequisite(s)	If the CSMS supports sending a SetDisplayMessageRequest with a transactionId for a transaction that does not exist.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: **State** is *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Request the CSMS to send a display message for a specific transaction.*	
	2. The OCTT responds with a **SetDisplayMessageResponse** with **status** *UnknownTransaction*	**1.** The CSMS sends a **SetDisplayMessageRequest**
Tool validations	* Step 1: Message **SetDisplayMessageRequest** - **message.transactionId** not *omit* AND - **message.priority** *<Configured Priority>*	
	Post scenario validations: - N/a	

Table 685. Test Case Id: TC_O_11_CSMS

Test case name	**Get a Specific Display Message - Unknown parameters**	
Test case Id	TC_O_11_CSMS	
Use case Id(s)	O04	
Requirement(s)	N/a	
System under test	CSMS	
Description	This test case describes how a CSO can request specific installed DisplayMessages configured via OCPP in a Charging Station. The Charging Station can remove messages when they are out-dated, or transactions have ended. It can be very useful for a CSO to be able to view to current list of messages, so the CSO knows which messages are (still) configured.	
Purpose	To verify if the CSMS is able to request a specific id message from the charging station according to the mechanism as described in the OCPP specification.	
Prerequisite(s)	If the CSMS is able to send a GetDisplayMessage with an unknown id.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: A display message is configured.	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **GetDisplayMessagesResponse** with **status** *Unknown*	**1.** The CSMS sends a **GetDisplayMessagesRequest**
Tool validations	* Step 1: Message **GetDisplayMessagesRequest** - **id** *<A different generated Id>* - **requestId** *<Generated Id>*	
	Post scenario validations: - N/a	

Table 686. Test Case Id: TC_O_12_CSMS

Test case name	**Set Display Message - Replace DisplayMessage**	
Test case Id	TC_O_12_CSMS	
Use case Id(s)	O06	
Requirement(s)	N/a	
System under test	CSMS	
Description	This test case describes how a CSO can replace a DisplayMessage that is previously configured in a Charging Station. Replace the message content, but also all the given parameters with the new one.	
Purpose	To verify if the CSMS is able to request to replace a display message according to the DisplayMessage mechanism as described in the OCPP specification.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: A display message is configured.	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	<u>Manual Action</u>: - *Request the CSMS to sent a display message with the same id as already configured one*	
	2. The OCTT responds with a **SetDisplayMessageResponse** with **status** *Accepted*	**1.** The CSMS sends a **SetDisplayMessageRequest** with: **message.id** *<Configured_Id>* **message.priority** *<Configured Priority>*
Tool validations	* Step 2: Message **SetDisplayMessageRequest** - **message.id** *<Configured_Id>* - **message.priority** *<Configured Priority>*	
	Post scenario validations: - N/a	

Table 687. Test Case Id: TC_O_13_CSMS

Test case name	**Set Display Message - Display message at StartTime**	
Test case Id	TC_O_13_CSMS	
Use case Id(s)	O01	
Requirement(s)	O01_FR_05	
System under test	CSMS	
Description	This test case describes how the CSMS can be requested to sent an SetDisplayMessageRequest to the charging station. Depending on the given parameters the message shall be displayed a certain way and at a certain moment on the Charging Station. These messages are displayed additionally on a Charging Station and are not part of the firmware.	
Purpose	To verify if the CSMS is able to send the request with a startTime according to the DisplayMessage mechanism as described in the OCPP specification.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Request the CSMS to send a SetDisplayMessageRequest with a startTime.*	
	2. The OCTT responds with a **SetDisplayMessageResponse** with **status** *Accepted*	**1.** The CSMS sends a **SetDisplayMessageRequest**
Tool validations	* Step 1: Message **SetDisplayMessageRequest** - **message.id** *<Generated Id>* - **message.startDateTime** *<Configured startDateTime>*	
	Post scenario validations: - N/a	

Table 688. Test Case Id: TC_O_14_CSMS

Test case name	**Set Display Message - Remove message after EndTime**	
Test case Id	TC_O_14_CSMS	
Use case Id(s)	O01	
Requirement(s)	O01_FR_05	
System under test	CSMS	
Description	This test case describes how the CSMS can be requested to sent an SetDisplayMessageRequest to the charging station. Depending on the given parameters the message shall be displayed a certain way and at a certain moment on the Charging Station. These messages are displayed additionally on a Charging Station and are not part of the firmware.	
Purpose	To verify if the CSMS is able to send the request with a endTime according to the DisplayMessage mechanism as described in the OCPP specification.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Request the CSMS to send a SetDisplayMessageRequest with a endTime.*	
	2. The OCTT responds with a **SetDisplayMessageResponse** with **status** *Accepted*	**1.** The CSMS sends a **SetDisplayMessageRequest**
Tool validations	* Step 1: Message **SetDisplayMessageRequest** - **message.id** *<Generated Id>* - **message.endDateTime** *<Configured endDateTime>*	
	Post scenario validations: - N/a	

Table 689. Test Case Id: TC_O_17_CSMS

Test case name	**Set Display Message - NotSupportedPriority**	
Test case Id	TC_O_17_CSMS	
Use case Id(s)	O01	
Requirement(s)	N/a	
System under test	CSMS	
Description	This test case describes how the CSMS can be requested to sent an SetDisplayMessageRequest to the charging station. Depending on the given parameters the message shall be displayed a certain way and at a certain moment on the Charging Station. These messages are displayed additionally on a Charging Station and are not part of the firmware.	
Purpose	To verify if the CSMS is able to send a display message with a specific priority, on which the Charging station responds not supported, according to the DisplayMessage mechanism as described in the OCPP specification.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **SetDisplayMessageResponse** with **status** *NotSupportedPriority*	**1.** The CSMS sends a **SetDisplayMessageRequest**
Tool validations	* Step 1: Message **SetDisplayMessageRequest** - **message.id** *<Generated Id>* - **message.priority** *<Configured priority>*	
	Post scenario validations: - N/a	

Table 690. Test Case Id: TC_O_18_CSMS

Test case name	**Set Display Message - NotSupportedState**	
Test case Id	TC_O_18_CSMS	
Use case Id(s)	O01	
Requirement(s)	N/a	
System under test	CSMS	
Description	This test case describes how the CSMS can be requested to sent an SetDisplayMessageRequest to the charging station. Depending on the given parameters the message shall be displayed a certain way and at a certain moment on the Charging Station. These messages are displayed additionally on a Charging Station and are not part of the firmware.	
Purpose	To verify if the CSMS is able to send a display message with a specific state, on which the Charging station responds not supported, according to the DisplayMessage mechanism as described in the OCPP specification.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **SetDisplayMessageResponse** with **status** *NotSupportedState*	**1.** The CSMS sends a **SetDisplayMessageRequest**
Tool validations	* Step 1: Message **SetDisplayMessageRequest** - **message.id** *<Generated Id>* - **message.state** *<Configured state>*	
	Post scenario validations: - N/a	

Table 691. Test Case Id: TC_O_19_CSMS

Test case name	**Set Display Message - NotSupportedMessageFormat**	
Test case Id	TC_O_19_CSMS	
Use case Id(s)	O01	
Requirement(s)	N/a	
System under test	CSMS	
Description	This test case describes how the CSMS can be requested to sent an SetDisplayMessageRequest to the charging station. Depending on the given parameters the message shall be displayed a certain way and at a certain moment on the Charging Station. These messages are displayed additionally on a Charging Station and are not part of the firmware.	
Purpose	To verify if the CSMS is able to send a display message with a specific MessageFormat, on which the Charging station responds not supported, according to the DisplayMessage mechanism as described in the OCPP specification.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **SetDisplayMessageResponse** with **status** *NotSupportedMessageFormat*	**1.** The CSMS sends a **SetDisplayMessageRequest**
Tool validations	* Step 1: Message **SetDisplayMessageRequest** - **message.id** *<Generated Id>*	
	Post scenario validations: - N/a	

Table 692. Test Case Id: TC_O_25_CSMS

Test case name	**Set Display Message - Send Specific state**	
Test case Id	TC_O_25_CSMS	
Use case Id(s)	O01	
Requirement(s)	N/a	
System under test	CSMS	
Description	This test case describes how the CSMS can be requested to sent an SetDisplayMessageRequest to the charging station. Depending on the given parameters the message shall be displayed a certain way and at a certain moment on the Charging Station. These messages are displayed additionally on a Charging Station and are not part of the firmware.	
Purpose	To verify if the CSMS is able to send a display messages with a "Charging" state according to the DisplayMessage mechanism as described in the OCPP specification.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **SetDisplayMessageResponse** with **status** *Accepted*	**1.** The CSMS sends a **SetDisplayMessageRequest**
Tool validations	* Step 1: Message **SetDisplayMessageRequest** - **message.id** *<Configured_Id>* - **message.state** *<Configured State>*	
	Post scenario validations: - N/a	

Table 693. Test Case Id: TC_O_26_CSMS

Test case name	**Set Display Message - Rejected**	
Test case Id	TC_O_26_CSMS	
Use case Id(s)	O01	
Requirement(s)	N/a	
System under test	CSMS	
Description	This test case describes how the CSMS can be requested to sent an SetDisplayMessageRequest to the charging station. Depending on the given parameters the message shall be displayed a certain way and at a certain moment on the Charging Station. These messages are displayed additionally on a Charging Station and are not part of the firmware.	
Purpose	To verify if the CSMS is able to send the request according to the DisplayMessage mechanism as described in the OCPP specification which gets rejected.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Request the CSMS to send a SetDisplayMessageRequest with a Normal Cycle priority.*	
	2. The OCTT responds with a **SetDisplayMessageResponse** with **status** *Rejected*	**1.** The CSMS sends a **SetDisplayMessageRequest**
Tool validations	* Step 1: Message **SetDisplayMessageRequest** - **message.id** *<Generated Id>* - **message.priority** *<Configured Priority>*	
	Post scenario validations: - N/a	

Table 694. Test Case Id: TC_O_27_CSMS

Test case name	**Set Display Message - Specific transaction - Display message at StartTime**	
Test case Id	TC_O_27_CSMS	
Use case Id(s)	O02	
Requirement(s)	N/a	
System under test	CSMS	
Description	This test case describes how the CSMS can be requested to sent an SetDisplayMessageRequest to the charging station. Depending on the given parameters the message shall be displayed a certain way and at a certain moment on the Charging Station. These messages are displayed additionally on a Charging Station and are not part of the firmware.	
Purpose	To verify if the CSMS is able to send the request with a startTime for a specific transaction according to the DisplayMessage mechanism as described in the OCPP specification.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a1	
	Charging State: **State** is *EnergyTransferStarted*	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **SetDisplayMessageResponse** with **status** *Accepted*	**1.** The CSMS sends a **SetDisplayMessageRequest**
Tool validations	* Step 1: Message **SetDisplayMessageRequest** - **message.id** *<Generated Id>* - **message.startDateTime** *<Configured startDateTime>* - **message.transactionId** is present	
	Post scenario validations: - N/a	

Table 695. Test Case Id: TC_O_28_CSMS

Test case name	**Set Display Message - Specific transaction - Remove message after EndTime**	
Test case Id	TC_O_28_CSMS	
Use case Id(s)	O02	
Requirement(s)	N/a	
System under test	CSMS	
Description	This test case describes how the CSMS can be requested to sent an SetDisplayMessageRequest to the charging station. Depending on the given parameters the message shall be displayed a certain way and at a certain moment on the Charging Station. These messages are displayed additionally on a Charging Station and are not part of the firmware.	
Purpose	To verify if the CSMS is able to send the request with a endTime for a specific transaction according to the DisplayMessage mechanism as described in the OCPP specification.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Charging State: N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	2. The OCTT responds with a **SetDisplayMessageResponse** with **status** *Accepted*	**1.** The CSMS sends a **SetDisplayMessageRequest**
Tool validations	* Step 1: Message **SetDisplayMessageRequest** - **message.id** *<Generated Id>* - **message.priority** *<Configured Priority>* - **message.endDateTime** *<Configured endDateTime>* - **message.state** *<Configured State>* - **message.transactionId** is present	
	Post scenario validations: - N/a	

3.17. P DataTransfer

Table 696. Test Case Id: TC_P_02_CSMS

Test case name	**Data Transfer to the CSMS - Rejected / Unknown VendorId / Unknown MessageId**	
Test case Id	TC_P_02_CSMS	
Use case Id(s)	P02	
Requirement(s)	P02.FR.06, P02.FR.07	
System under test	CSMS	
Description	The DataTransfer message to send information for functions that are not supported by OCPP.	
Purpose	To verify whether the CSMS is able to handle receiving a DataTransferRequest, even if it does not support any vendor-specific implementations.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The OCTT sends a **DataTransferRequest** with **vendorId** *<Configured vendorId>* **messageId** *<Configured messageId>*	**2.** The CSMS responds with a **DataTransferResponse**
Tool validations	* Step 2: Message: **DataTransferResponse** - **status** must be *UnknownVendorId* OR *UnknownMessageId* OR *Rejected* (Rejected will also be allowed, because there are implementers that like to just reject the message when the Charging Station does not support any vendor-specific features.	
	Post scenario validations: N/a	

Table 697. Test Case Id: TC_P_03_CSMS

Test case name	**CustomData - Receive custom data**	
Test case Id	TC_P_03_CSMS	
Use case Id(s)	N/a	
Requirement(s)	N/a	
System under test	CSMS	
Description	Checks if the CSMS is able to receive custom data.	
Purpose	To verify whether the CSMS is able to handle receiving custom data.	
Prerequisite(s)	N/a	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	1. The OCTT sends a **StatusNotificationRequest** with **customData** *<customData>*	**2.** The CSMS responds with a **StatusNotificationResponse**
	3. The OCTT sends a **TransactionEventRequest** with **customData** *customData* **transactionInfo.customData** *<customData>*	**4.** The CSMS responds with a **TransactionEventResponse**
Tool validations	N/a	
	Post scenario validations: N/a	

3.18. Reusable states

Testcases can refer to a reusable state at the before or main stage. The steps described at the reusable state will be executed and then it will return to the testcase that called the reusable state.

Table 698. Reusable State: Booted

State	**Booted**	
System under test	CSMS	
Description	This state will simulate that the Charging Station is completely power cycled. The OCTT end in a state where it is "booted" back up and is in idle mode.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Scenario)	**Charging Station**	**CSMS**
	1. The OCTT sends a **BootNotificationRequest** with **reason** *PowerUp* **chargingStation.model** *<Configured model>* **chargingStation.vendorName** *<Configured vendorName>*	**2.** The CSMS responds with a **BootNotificationResponse**
	3. The OCTT notifies the CSMS about the current state of all connectors. Message: **StatusNotificationRequest** with **connectorStatus** *Available* Message: **NotifyEventRequest** with **trigger** *Delta* **actualValue** *"Available"* **component.name** *"Connector"* **variable.name** *"AvailabilityState"*	**4.** The CSMS responds accordingly.
Tool validations	* Step 2: Message: **BootNotificationResponse** - **status** *Accepted*	
Post condition	**State** is *Booted*	

Table 699. Reusable State: Reserved

State	**Reserved**	
System under test	CSMS	
Description	This state will simulate a reservation for a specified evse.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Scenario)	**Charging Station**	**CSMS**
	Manual Action: *Trigger the CSMS to send a ReserveNowRequest for specific EVSE.*	
	2. The OCTT responds with a **ReserveNowResponse** With status *Accepted*	**1.** The CSMS sends a **ReserveNowRequest**
	3. The OCTT notifies the CSMS about the current state of the connector(s) of the Specified EVSE Message: **StatusNotificationRequest** with **connectorStatus** *Reserved* Message: **NotifyEventRequest** with **trigger** *Delta* **actualValue** *"Reserved"* **component.name** *"Connector"* **variable.name** *"AvailabilityState"*	**4.** The CSMS responds accordingly.
Tool validations	* Step 1: Message: **ReserveNowRequest** - **evseId** must be *<Specified evseId>* - **connectorType** must be omitted - **idToken.idToken** *<Configured valid_idtoken_idtoken>* - **idToken.type** *<Configured valid_idtoken_type>*	
Post condition	**State** is *Reserved*	

Table 700. Reusable State: Unavailable

State	**Unavailable**	
System under test	CSMS	
Description	This state will simulate that Charging Station / EVSEs / connectors are set to AvailabilityState Unavailable.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Scenario)	**Charging Station**	**CSMS**
	Manual Action: *Request the CSMS to change the availability of the specified components to Inoperative.*	
	2. The OCTT responds with a **ChangeAvailabilityResponse** with **status** *Accepted*	**1.** The CSMS sends a **ChangeAvailabilityRequest**
	3. The OCTT notifies the CSMS about the current state of all connectors belonging to the specified EVSE (and optionally also from the EVSE itself). Message: **StatusNotificationRequest** - **connectorStatus** *Unavailable* Message: **NotifyEventRequest** - **trigger** *Delta* - **actualValue** *"Unavailable"* - **component.name** *"ChargingStation" / EVSE / Connector* - **variable.name** *"AvailabilityState"*	**4.** The CSMS responds accordingly.
Tool validations	* Step 1: Message **ChangeAvailabilityRequest** - **operationalStatus** *Inoperative* - **evse** *<Specified evseId>* - **connectorId** *omitted*	
Post condition	**State** is *Unavailable*	

Table 701. Reusable State: EVConnectedPreSession

State	**EVConnectedPreSession**	
System under test	CSMS	
Description	This state will simulate that the EV and EVSE of the simulated Charging Station are connected.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Scenario)	**Charging Station**	**CSMS**
	1. The OCTT notifies the CSMS about the status change of the connector Message: **StatusNotificationRequest** - **connectorStatus** is *Occupied* Message: **NotifyEventRequest** - **trigger** is *Delta* - **actualValue** is *Occupied* - **component.name** is *Connector* - **variable.name** is *AvailabilityState*	**2.** The CSMS responds accordingly.
	3. The OCTT sends a **TransactionEventRequest** With **triggerReason** is *CablePluggedIn* **transactionInfo.chargingState** is *EVConnected* **evse.id** *<Configured evseId>* **evse.connectorId** *<Configured connectorId>* If **State** is *Authorized* then **eventType** is *Updated* else **eventType** is *Started*	**4.** The CSMS responds with a **TransactionEventResponse**
Tool validations	N/a	
Post condition	**State** is *EVConnectedPreSession*	

Table 702. Reusable State: Authorized

State	**Authorized**	
System under test	CSMS	
Description	This state will simulate that the EV Driver is locally authorizing to start a transaction on the simulated Charging Station.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Scenario)	**Charging Station**	**CSMS**
	1. The OCTT sends an **AuthorizeRequest** With **idToken.idToken** *<Configured valid_idtoken_idtoken>* **idToken.type** *<Configured valid_idtoken_type>*	**2.** The CSMS responds with an **AuthorizeResponse**
	3. The OCTT sends a **TransactionEventRequest** With **triggerReason** is *Authorized* **idToken.idToken** *<Configured valid_idtoken_idtoken>* **idToken.type** *<Configured valid_idtoken_type>* If **State** is *EVConnectedPreSession* then **eventType** is *Updated* else **eventType** is *Started*	**4.** The CSMS responds with a **TransactionEventResponse**
Tool validations	* Step 2: Message: **AuthorizeResponse** - **idTokenInfo.status** must be *Accepted* * Step 4: Message: **TransactionEventResponse** - **idTokenInfo.status** must be *Accepted*	
Post condition	**State** is *Authorized*	

Table 703. Reusable State: EnergyTransferStarted

State	**EnergyTransferStarted**	
System under test	CSMS	
Description	This state will simulate that there is transferring energy between the EV and EVSE of the simulated Charging Station.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): If **State** is NOT *Authorized* then execute **Reusable State** *Authorized* If **EVConnected** is *true*, then proceed to part 2 Else proceed to part 1.	
Main (Part 1) (Scenario)	**Charging Station**	**CSMS**
	1. The OCTT notifies the CSMS about the status change of the connector. Message: **StatusNotificationRequest** - **connectorStatus** is *Occupied* Message: **NotifyEventRequest** - **trigger** is *Delta* - **actualValue** is *Occupied* - **component.name** is *Connector* - **variable.name** is *AvailabilityState*	**2.** The CSMS responds accordingly.
	3. The OCTT sends a **TransactionEventRequest** With **triggerReason** is *CablePluggedIn* **transactionInfo.chargingState** is *EVConnected* evse.id *<Configured evseId>* evse.connectorId *<Configured connectorId>* **eventType** is *Updated*	**4.** The CSMS responds with a **TransactionEventResponse**
Tool validations	N/a	
Main (Part 2) (Scenario)	**Charging Station**	**CSMS**
	5. The OCTT sends a **TransactionEventRequest** With **triggerReason** is *ChargingStateChanged* **transactionInfo.chargingState** is *Charging* **eventType** is *Updated*	**6.** The CSMS responds with a **TransactionEventResponse**
Tool validations	N/a	
Post condition	**State** is *EnergyTransferStarted* **EVConnected** is *true*	

Table 704. Reusable State: EnergyTransferSuspended

State	**EnergyTransferSuspended**	
System under test	CSMS	
Description	This state will simulate that the Charging Station is in a state where the energy transfer is suspended by the EV.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): If **State** is NOT *EnergyTransferStarted* then execute **Reusable State** *EnergyTransferStarted*	
Main (Scenario)	**Charging Station**	**CSMS**
	Notes(s): *The tool will wait for <Configured Transaction Duration> seconds*	
	1. The OCTT sends a **TransactionEventRequest** With **triggerReason** is *ChargingStateChanged* **transactionInfo.chargingState** is *SuspendedEV*	**2.** The CSMS responds with a **TransactionEventResponse**
Tool validations	N/a	
Post condition	**State** is *EnergyTransferSuspended*	

Table 705. Reusable State: StopAuthorized

State	**StopAuthorized**	
System under test	CSMS	
Description	This state will simulate that the Charging Station is in a state where the charging session is authorized to stop.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): If **State** is NOT *EnergyTransferStarted* then execute **Reusable State** *EnergyTransferStarted*	
Main (Scenario)	**Charging Station**	**CSMS**
	Notes(s): *The tool will wait for <Configured Transaction Duration> seconds*	
	1. The OCTT sends a **TransactionEventRequest** With **triggerReason** is *StopAuthorized* **eventType** is *Updated*	**2.** The CSMS responds with a **TransactionEventResponse**
Tool validations	* Step 2: Message: **TransactionEventResponse** - **idTokenInfo.status** must be *Accepted*	
Post condition	**State** is *StopAuthorized*	

Table 706. Reusable State: EVConnectedPostSession

State	**EVConnectedPostSession**	
System under test	CSMS	
Description	This state will simulate that the Charging Station is in a state where the energy transfer has been stopped and the transaction is NOT authorized to resume energy transfer without re-authorization.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): If **State** is NOT *StopAuthorized* then execute **Reusable State** *StopAuthorized*	
Main (Scenario)	**Charging Station**	**CSMS**
	1. The OCTT sends a **TransactionEventRequest** With **triggerReason** is *ChargingStateChanged* **transactionInfo.chargingState** is *EVConnected* **eventType** is *Updated*	**2.** The CSMS responds with a **TransactionEventResponse**
Tool validations	N/a	
Post condition	**State** is *EVConnectedPostSession*	

Table 707. Reusable State: EVDisconnected

State	**EVDisconnected**	
System under test	CSMS	
Description	This state will simulate that the EV and EVSE of the simulated Charging Station are disconnected, after the charging session is authorized to stop.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): If **State** is NOT *EVConnectedPostSession* then execute **Reusable State** *EVConnectedPostSession*	
Main (Scenario)	**Charging Station**	**CSMS**
	1. The OCTT notifies the CSMS about the status change of the connector. Message: **StatusNotificationRequest** - **connectorStatus** is *Available* Message: **NotifyEventRequest** - **trigger** is *Delta* - **actualValue** is *Available* - **component.name** is *Connector* - **variable.name** is *AvailabilityState*	**2.** The CSMS responds accordingly.
	3. The OCTT sends a **TransactionEventRequest** With **triggerReason** is *EVCommunicationLost* **transactionInfo.chargingState** is *Idle* **transactionInfo.stoppedReason** is *EVDisconnected* **eventType** is *Ended*	**4.** The CSMS responds with a **TransactionEventResponse**
Tool validations	N/a	
Post condition	**State** is *EVDisconnected*	

Table 708. Reusable State: GetInstalledCertificates

State	**GetInstalledCertificates**	
System under test	CSMS	
Description	The hashData from installed certificates of the specified type will be retrieved from the Charging Station	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Scenario)	**Charging Station**	**CSMS**
	Manual Action: *Trigger the CSMS to send a GetInstalledCertificateIdsRequest with certificateType _<Specified certificateType>*	
	2. The OCTT responds with a **GetInstalledCertificateIdsResponse** With **status** is *Accepted* **certificateHashDataChain** contains an entry with following values: **certificateHashDataChain[0].certificateType** is *<Specified certificateType>* **certificateHashDataChain[0].certificateHashData** contains *<HashData from the configured certificate of the specified certificateType>*	**1.** The CSMS sends a **GetInstalledCertificateIdsRequest**
Tool validations	* Step 1: Message: **GetInstalledCertificateIdsRequest** - **certificateType** must be *<Specified certificateType>*	
Post condition	Certificate of the specified certificateType is retrieved from the Charging Station.	

Table 709. Reusable State: CertificateInstalled

State	**CertificateInstalled**	
System under test	CSMS	
Description	A pre configured certificate of the specified certificateType will be installed.	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Scenario)	**Charging Station**	**CSMS**
	Manual Action: *Trigger the CSMS to send an InstallCertificateRequest with certificateType <Specified certificateType>*	
	2. The OCTT responds with a **InstallCertificateResponse** With **status** is *Accepted*	**1.** The CSMS sends a **InstallCertificateRequest**
Tool validations	* Step 1: Message: **InstallCertificateRequest** - **certificateType** must be *<Specified certificateType>* - **certificate** must be *<The configured certificate of the specified certificateType.>*	
Post condition	Certificate of the specified certificateType is stored at the Charging Station.	

Table 710. Reusable State: ISO15118SmartCharging

State	**ISO15118SmartCharging**	
System under test	CSMS	
Description		
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Scenario)	**Charging Station**	**CSMS**
	1. The OCTT sends a **NotifyEVChargingNeedsRequest** with **evseId** *<Configured evseId>* **maxScheduleTuples** & **chargingNeeds** *<Configured values from mock EV>+*	**2.** The CSMS responds with a **NotifyEVChargingNeedsResponse**.
	4. The OCTT responds with a **SetChargingProfileResponse** with: **status** *Accepted*	**3.** The CSMS sends a **SetChargingProfileRequest** Note(s): *- If **NotifyEVChargingNeedsResponseStatus** was Processing, the OCTT will wait 60 seconds for the request*
	5. The OCTT sends a **NotifyEVChargingScheduleRequest** with **evseId** *<COnfigured evseId>* **chargingSchedule** *<ChargingSchedule provided at step 3>*	**6.** The CSMS responds with a **NotifyEVChargingScheduleResponse**.
	7. The OCTT sends a **TransactionEventRequest** with **triggerReason** *<ChargingStateChanged>* **transactionInfo.chargingState** *<Charging>*	**8.** The CSMS responds with a **TransactionEventResponse**.

State	**ISO15118SmartCharging**
Tool validations	* Step 2: Message: **NotifyEVChargingNeedsResponse** - **Status** *Accepted* or *Processing* * Step 3: Message: **SetChargingProfileRequest** - **chargingProfilePurpose** *<TxProfile>* - **transactionId** *<Provided transactionId from before>* * Step 4: Message: **NotifyEVChargingScheduleResponse** - **status** *<Accepted>*
Post condition	N/a

Table 711. Memory State: RenewChargingStationCertificate

State	**RenewChargingStationCertificate**	
System under test	CSMS	
Description	The ChargingStationCertificate is renewed using A02/A03	
Before (Preparations)	**Configuration State:** N/a	
	Memory State: N/a	
	Reusable State(s): N/a	
Main (Test scenario)	**Charging Station**	**CSMS**
	Manual Action: *Request the CSMS to send a Trigger Message Request with **requestedMessage** SignChargingStationCertificate*	
	2. The OCTT sends a **TriggerMessageResponse** with **status** *Accepted*	**1.** The CSMS sends a **TriggerMessageRequest** With **requestedMessage** *SignChargingStationCertificate*
	3 The OCTT sends a **SignCertificateRequest**	**4.** The CSMS responds with a **SignCertificateResponse** with **status** *Accepted*
	6. The OCTT sends a **CertificateSignedResponse** with **status** *Accepted*	**5.** The CSMS sends a **CertificateSignedRequest** With **certificateChain** *<Certificate generated from the received CSR from step 3 and signed by the configured CSMS Root certificate>* **certificateType** *ChargingStationCertificate*
Tool validations	* Step 1: Message: **TriggerMessageRequest** - **requestedMessage** must be *SignChargingStationCertificate* * Step 4: Message: **SignCertificateResponse** - **status** must be *Accepted* * Step 5: Message: **CertificateSignedRequest** - **certificateChain** *<Certificate generated from the received CSR from step 3 and signed by the configured CSMS Root certificate>* - **certificateType** must be *ChargingStationCertificate*	
	Post scenario validations: N/a	