



OCPP 2.1
Part 6 - Test Cases

Edition 2, 2025-12-03

Table of Contents

1. Introduction	2
About this document	2
Conventions	2
2. Test Cases Charging Station	3
A Security	4
B Provisioning	28
C Authorization	114
D Local Authorization List Management	224
E Transactions	233
F Remote Control	317
G Availability	344
H Reservation	369
I Tariff and Cost	398
J Meter Values	461
K Smart Charging	473
L Firmware Management	598
M CertificateManagement	638
N Diagnostics	667
O Display Message	759
P Data Transfer	806
Q Bidirectional Power Transfer	809
R DER Control	879
S Battery Swapping	901
Memory states	906
Reusable states	915
3. Test Cases Charging Station Management System	955
A Security	956
B Provisioning	976
C Authorization	1009
D Local Authorization List Management	1050
E Transactions	1056
F Remote Control	1102
G Availability	1119
H Reservation	1129
I Tariff and Cost	1142
J Meter Values	1159
K Smart Charging	1169
L Firmware Management	1222
M Certificate Management	1248
N Diagnostics	1269
O Display Message	1311
P Data Transfer	1334
Q Bidirectional Power Transfer	1336
R DER Control	1359
S Battery Swapping	1364
Reusable states	1368

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

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

Version History

Version	Date	Modified by	Description
OCPP 2.1 Edition 2	2025-12-03	Open Charge Alliance	OCPP 2.1 Edition 2. This is the first published version of OCPP 2.1 Part 6.

1. Introduction

About this document

This document is created to describe a set of valid test cases for OCPP 2.1. These test cases can be executed using a Test System for OCPP 2.1. 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. Certificate Management
- N. Diagnostics
- O. Display Message
- P. Data Transfer
- Q. Bidirectional Power Transfer
- R. DER Control
- S. Battery Swapping

The scenarios in this document are also part of the OCA certification process of OCPP. Please refer to OCPP 2.1 Part 5 - Certification Profiles for more information about the relation between certification profiles and the test scenarios in this document.

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 2.1 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

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 Test System 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 Test System 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.

A Security

TC_A_01_CS: Basic Authentication - Valid username/password combination

Test case name	Basic Authentication - Valid username/password combination
Test case Id	TC_A_01_CS
Use case Id(s)	A00, B01
Requirement(s)	A00.FR.202, A00.FR.203, A00.FR.204, A00.FR.205, A00.FR.301, A00.FR.302, A00.FR.304 AND B01.FR.01, B01.FR.05, B01.FR.09
System under test	Charging Station
Description	The Charging Station uses Basic authentication to authenticate itself to the CSMS, when using security profile 1 or 2.
Purpose	To verify whether the Charging Station is able to authenticate itself to the CSMS using Basic Authentication.
Prerequisite(s)	<ul style="list-style-type: none"> - The charging station supports security profile 1 and/or 2 - The active NetworkConnectionProfile uses either security profile 1 OR 2.

Before (Preparations)
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 <i>Booted</i>	

Tool validations
<p>* Step 1:</p> <p>The authorization header of the HTTP upgrade request must be formatted as follows: AUTHORIZATION: Basic <Base64 encoded(<ChargingStationId>:<Configured basicAuthPassword>)></p> <ul style="list-style-type: none"> - The ChargingStationId, must equal the ChargingStationId provided at the end of the connection url string of the HTTP request. - BasicAuthPassword must consist of minimum 16 and maximum 40 characters - BasicAuthPassword may only contain alpha-numeric characters and the special characters allowed by passwordString.
Post scenario validations: N/a

TC_A_04_CS: TLS - server-side certificate - Valid certificate

Test case name	TLS - server-side certificate - Valid certificate
Test case Id	TC_A_04_CS
Use case Id(s)	A00
Requirement(s)	A00.FR.309,A00.FR.312,A00.FR.313,A00.FR.319,A00.FR.321,A00.FR.412,A00.FR.422
System under test	Charging Station
Description	The CSMS uses a server-side certificate to identify itself to the Charging Station, when using security profile 2 or 3.
Purpose	To verify whether the Charging Station is able to receive a server certificate provided by the CSMS and setup a secured WebSocket connection.
Prerequisite(s)	<ul style="list-style-type: none"> - The charging station supports security profile 2 and/or 3 - The active NetworkConnectionProfile uses either security profile 2 OR 3.

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>Booting</i>

Main (Test scenario)	
Charging Station	CSMS
1. The Charging Station initiates a TLS handshake and sends a Client Hello to the Test System.	2. The Test System 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 <u>Note(s):</u> - The client certificate is only sent when the Charging Station uses security profile 3.	4. The Test System performs the following actions: Change Cipher Spec Finished
5. The Charging Station sends a HTTP upgrade request to the Test System <u>Note(s):</u> - The HTTP request only contains a username/password combination when the Charging Station uses security profile 2.	6. The Test System upgrades the connection to a (secured) WebSocket connection.
7. The Charging Station sends a BootNotificationRequest	8. The Test System responds with a BootNotificationResponse with status Accepted
9. The Charging Station notifies the CSMS about the current state of all connectors.	10. The Test System responds accordingly.

Tool validations

* Step 2:
The Test System 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

TC_A_05_CS: TLS - server-side certificate - Invalid certificate

Test case name	TLS - server-side certificate - Invalid certificate
Test case Id	TC_A_05_CS
Use case Id(s)	A00
Requirement(s)	A00.FR.309,A00.FR.310,A00.FR.311,A00.FR.412,A00.FR.413,A00.FR.414
System under test	Charging Station
Description	The CSMS uses a server-side certificate to identify itself to the Charging Station, when using security profile 2 or 3.
Purpose	To verify whether the Charging Station is able to terminate the connection when the received server certificate is invalid.
Prerequisite(s)	<ul style="list-style-type: none"> - The charging station supports security profile 2 and/or 3 - The active NetworkConnectionProfile uses either security profile 2 OR 3.

Before (Preparations)
Configuration State: OCPPCommCtrlr.NetworkProfileConnectionAttempts is 3 OCPPCommCtrlr.NetworkConfigurationPriority only contains <Value from ActiveNetworkProfile> SecurityCtrlr.AllowCSMSTLSWildcards is false (If implemented)
Memory State: N/a
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
1. The Test System aborts the connection with the Charging Station.	
2. The Charging Station initiates a TLS handshake and sends a Client Hello to the Test System.	3. The Test System responds with a Server Hello With a <Configured valid server certificate> <u>Note(s):</u> - The Test System will use this as an indication of the time it takes the Charging Station to reconnect.
4. The Test System aborts the connection with the Charging Station.	
5. The Charging Station initiates a TLS handshake and sends a Client Hello to the Test System.	6. The Test System responds with a Server Hello With a <Generated invalid server certificate>
7. The Charging Station deems the server certificate invalid and terminates the connection.	
<u>Note:</u> The Test System will wait two times the measured reconnection time from step 3, before switching the server certificate back to the valid server certificate. The reason for this is that the Test System is not always able to detect a failed connection attempt.	
8. The Charging Station initiates a TLS handshake and sends a Client Hello to the Test System.	9. The Test System responds with a Server Hello With a <Configured valid server certificate> <u>Note(s):</u> - The Test System will accept the connection to prevent doubling of the RetryBackOffWaitMinimum.
10 The Charging Station sends a SecurityEventNotificationRequest	11 The Test System responds with a SecurityEventNotificationResponse

Main (Test scenario)Note(s):

The Test System will loop through steps 4 to 11 for a set of generated invalid certificates;
"Expired", "Future validity date", "Not signed by installed CSMS Root certificate", "CommonName that does not equal the FQDN of the server", "CommonName containing a wildcard hostname matching the FQDN".

Tool validations

* Step 10:

Message: **SecurityEventNotificationRequest**

- **type** must be *InvalidCsmsCertificate*

Post scenario validations:

N/a

TC_A_06_CS: TLS - server-side certificate - TLS version too low

Test case name	TLS - server-side certificate - TLS version too low
Test case Id	TC_A_06_CS
Use case Id(s)	A00
Requirement(s)	A00.FR.314,A00.FR.316,A00.FR.416,A00.FR.417,A00.FR.419
System under test	Charging Station
Description	The CSMS uses a server-side certificate to identify itself to the Charging Station, when using security profile 2 or 3.
Purpose	To verify whether the Charging Station is able to terminate the connection when it notices the used TLS version is lower than 1.2.
Prerequisite(s)	<ul style="list-style-type: none"> - The charging station supports security profile 2 and/or 3 - The active NetworkConnectionProfile uses either security profile 2 OR 3.

Before (Preparations)
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 Test System.	2. The Test System 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 Test System.	5. The Test System responds with a Server Hello With the <Configured server certificate>
6. The Charging Station performs the following actions: Send client certificate Client Key Exchange Certificate verify Change Cipher Spec Finished <u>Note(s):</u> - The client certificate is only sent when the Charging Station uses security profile 3.	7. The Test System performs the following actions: Change Cipher Spec Finished
8. The Charging Station sends a HTTP upgrade request to the Test System <u>Note(s):</u> - The HTTP request only contains a username/password combination when the Charging Station uses security profile 2.	9. The Test System upgrades the connection to a (secured) WebSocket connection.
10. The Charging Station sends a BootNotificationRequest	11. The Test System responds with a BootNotificationResponse with status Accepted
12. The Charging Station notifies the CSMS about the current state of all connectors.	13. The Test System responds accordingly.

Main (Test scenario)	
14 The Charging Station sends a SecurityEventNotificationRequest	15 The Test System responds with a SecurityEventNotificationResponse
16 The Charging Station sends a SecurityEventNotificationRequest	17 The Test System responds with a SecurityEventNotificationResponse
<u>Note(s):</u> - 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 <i>StartupOfTheDevice</i> or <i>ResetOrReboot</i>
* Step 16: Message: SecurityEventNotificationRequest - type must be <i>InvalidTLSVersion</i>
Post scenario validations: N/a

TC_A_07_CS: TLS - Client-side certificate - valid certificate

Test case name	TLS - Client-side certificate - valid certificate
Test case Id	TC_A_07_CS
Use case Id(s)	A00
Requirement(s)	A00.FR.401,A00.FR.402,A00.FR.415,A00.FR.416,A00.FR.422,A00.FR.502,A00.FR.503,A00.FR.507,A00.FR.508,A00.FR.511
System under test	Charging Station
Description	The Charging Station uses a client-side certificate to identify itself to the CSMS, when using security profile 3.
Purpose	To verify whether the Charging Station is able to provide a valid client certificate and setup a secured WebSocket connection.
Prerequisite(s)	<ul style="list-style-type: none"> - The charging station supports security profile 3 - The active NetworkConnectionProfile uses security profile 3.

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>Booting</i>

Main (Test scenario)	
Charging Station	CSMS
1. The Charging Station initiates a TLS handshake and sends a Client Hello to the Test System.	2. The Test System 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 Test System performs the following actions: Change Cipher Spec Finished
5. The Charging Station sends a HTTP upgrade request to the Test System	6. The Test System upgrades the connection to a (secured) WebSocket connection.
7. The Charging Station sends a BootNotificationRequest	8. The Test System responds with a BootNotificationResponse with status Accepted
9. The Charging Station notifies the CSMS about the current state of all connectors.	10. The Test System responds accordingly.

Tool validations
<p>* Step 4:</p> <p>The Test System validates the following before finishing the TLS handshake:</p> <ul style="list-style-type: none"> - The Charging Station must use TLS version 1.2 or above <p>At least the following set of cipher suites must be supported:</p> <p>(TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256</p> <p>AND</p> <p>TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384)</p> <p>OR</p> <p>(TLS_RSA_WITH_AES_128_GCM_SHA256</p> <p>AND</p> <p>TLS_RSA_WITH_AES_256_GCM_SHA384)</p> <ul style="list-style-type: none"> - When using RSA or DSA the key must be at least 2048 bits long. <p>and when using elliptic curve cryptography the key must be at least 224 bits long.</p> <ul style="list-style-type: none"> - The received Client side certificate must be transmitted in the X.509 format encoded in Privacy-Enhanced Mail (PEM) format. - The certificate must include a serial number. - The subject field of the certificate must contain a commonName RDN which consists of the unique serial number of the Charging Station. <p><i>NOTE: If one of the above validations fails, the Test System can still setup the WebSocket connection (if it is able to), but the testcase will FAIL and the Test System reports why it failed.</i></p> <p>* Step 9:</p> <p>Message: StatusNotificationRequest</p> <ul style="list-style-type: none"> - connectorStatus <i>Available</i> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none"> - eventData[0].trigger <i>Delta</i> - eventData[0].actualValue <i>"Available"</i> - eventData[0].component.name <i>"Connector"</i> - eventData[0].variable.name <i>"AvailabilityState"</i> <p>Post scenario validations:</p> <p>N/a</p>

TC_A_09_CS: Update Charging Station Password for HTTP Basic Authentication - Accepted

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetVariablesResponse	1. The Test System 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 <i>BasicAuthPassword</i>). <u>Note(s):</u> - The Authorization header is formatted as follows: <i>AUTHORIZATION: Basic <Base64 encoded(<Configured ChargingStationId>:<NEW BasicAuthPassword>)></i>	4. The Test System validates the username/password combination AND upgrades the connection to a (secured) WebSocket connection.
5. The Charging Station sends a BootNotificationRequest	6. The Test System responds with a BootNotificationResponse
7. The Charging Station notifies the Test System about the current state of all connectors.	8. The Test System responds accordingly.
<u>Note(s):</u> - Steps 5, 6, 7, and 8 are only required when status in Step 2 is <i>RebootRequired</i>	

Tool validations
* Step 2: Message: SetVariablesResponse - status must be <i>Accepted</i> or <i>RebootRequired</i>
Post scenario validations: N/a

TC_A_10_CS: Update Charging Station Password for HTTP Basic Authentication - Rejected

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetVariablesResponse	<p>1. The Test System sends a SetVariablesRequest</p> <p>setVariableData[1]:</p> <ul style="list-style-type: none"> - variable.name = "BasicAuthPassword" - component.name = "SecurityCtrlr" - attributeValue = "<NewPassword which is less than 16 characters>"
	<p>3. The ChargingStation sends a HTTP upgrade request with an Authorization header, containing a username/password combination (with the old <i>BasicAuthPassword</i>).</p> <p><u>Note(s):</u></p> <ul style="list-style-type: none"> - The Authorization header is formatted as follows: AUTHORIZATION: Basic <Base64 encoded(<Configured ChargingStationId>:<OLD BasicAuthPassword>)>
4. The Test System validates the username/password combination AND upgrades the connection to a (secured) WebSocket connection.	
5. Execute Reusable State <i>Booted</i>	

Tool validations
<p>* Step 2:</p> <p>Message: SetVariablesResponse</p> <ul style="list-style-type: none"> - status must be <i>Rejected</i>
<p>Post scenario validations:</p> <p>BasicAuthPassword should be <Configured BasicAuthPassword></p> <p>N/a</p>

TC_A_11_CS: Update Charging Station Certificate by request of CSMS - Success - Charging Station Certificate

Test case name	Update Charging Station Certificate by request of CSMS - Success - Charging Station Certificate
Test case Id	TC_A_11_CS
Use case Id(s)	A02 & F06
Requirement(s)	A02.FR.02, A02.FR.03, A02.FR.06, A02.FR.08, A02.FR.09 & F06.FR.04,F06.FR.05,F06.FR.10
System under test	Charging Station
Description	The CSMS is able to request the Charging Station to update its charging station certificate using the TriggerMessageRequest message.
Purpose	To verify if the Charging Station is able to update its Charging Station Certificate.
Prerequisite(s)	<ul style="list-style-type: none"> - The charging station supports security profile 3 - The active NetworkConnectionProfile uses security profile 3.

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

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>RenewChargingStationCertificate</i> for certificateType <i>ChargingStationCertificate</i>	

Tool validations
N/a
Post scenario validations: N/a

TC_A_12_CS: Update Charging Station Certificate by request of CSMS - Success - V2G Certificate

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 <i>RenewV2GChargingStationCertificate</i>	

Tool validations
N/a
Post scenario validations: N/a

TC_A_14_CS: Update Charging Station Certificate by request of CSMS - Invalid certificate

Test case name	Update Charging Station Certificate by request of CSMS - Invalid certificate
Test case Id	TC_A_14_CS
Use case Id(s)	A02
Requirement(s)	A02.FR.07,A03.FR.07
System under test	Charging Station
Description	The CSMS is able to request the Charging Station to update its charging station certificate using the TriggerMessageRequest message.
Purpose	To verify if the Charging Station is able to discard an invalid certificate and report a security event.
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station supports security profile 3 - The active NetworkConnectionProfile uses security profile 3.

Before (Preparations)
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 Test System sends a TriggerMessageRequest With requestedMessage <i>SignChargingStationCertificate</i>
3 The Charging Station sends a SignCertificateRequest	4. The Test System responds with a SignCertificateResponse With status <i>Accepted</i>
6. The Charging Station responds with a CertificateSignedResponse	5. The Test System sends a CertificateSignedRequest With certificateChain <i><Configured invalid_signingCertificate></i> certificateType <i>ChargingStationCertificate</i>
7 The Charging Station sends a SecurityEventNotificationRequest	8 The Test System responds with a SecurityEventNotificationResponse

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

TC_A_15_CS: Update Charging Station Certificate by request of CSMS - SignCertificateRequest Rejected

Test case name	Update Charging Station Certificate by request of CSMS - SignCertificateRequest Rejected
Test case Id	TC_A_15_CS
Use case Id(s)	A02
Requirement(s)	N/a
System under test	Charging Station
Description	The CSMS is able to request the Charging Station to update its charging station certificate using the TriggerMessageRequest message.
Purpose	To verify if the Charging Station is able to discard an invalid certificate and report a security event.
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station supports security profile 3 - The active NetworkConnectionProfile uses security profile 3.

Before (Preparations)
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 Test System sends a TriggerMessageRequest With requestedMessage <i>SignChargingStationCertificate</i>
3 The Charging Station sends a SignCertificateRequest	4. The Test System responds with a SignCertificateResponse With status <i>Rejected</i>

Tool validations
* Step 2: Message: TriggerMessageResponse - status must be <i>Accepted</i>
Post scenario validations: N/a

TC_A_23_CS: Update Charging Station Certificate by request of CSMS - CertificateSignedRequest Timeout

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. CSMS will after a delay send a CertificateSignedRequest for each SignCertificateRequest that it has accepted.
Prerequisite(s)	<ul style="list-style-type: none"> - The charging station supports security profile 3 - The Charging Station supports the CertificateSignedRequest Timeout feature

Before (Preparations)
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 Test System sends a TriggerMessageRequest With requestedMessage <i>SignChargingStationCertificate</i>
3 The Charging Station sends a SignCertificateRequest	4. The Test System responds with a SignCertificateResponse With status <i>Accepted</i>
6 The Charging Station sends a SignCertificateRequest	5. The Test System does NOT send the CertificateSignedRequest and waits for the SignCertificateRequest to be resend after the <Configured CertSigningWaitMinimum>
	7. The Test System responds with a SignCertificateResponse With status <i>Accepted</i>
9 The Charging Station sends a SignCertificateRequest	8. The Test System does NOT send the CertificateSignedRequest and waits for the SignCertificateRequest to be resend after the <Configured CertSigningWaitMinimum> times 2
	10. The Test System responds with a SignCertificateResponse With status <i>Accepted</i>
12. The Charging Station responds with a CertificateSignedResponse	11. The Test System sends a CertificateSignedRequest With certificateChain <Certificate generated from the received CSR from step 3 and signed by the provided CSMS Root certificate> certificateType <i>ChargingStationCertificate</i>

Main (Test scenario)	
14. The Charging Station responds with a CertificateSignedResponse	13. The Test System sends a CertificateSignedRequest With certificateChain <Certificate generated from the received CSR from step 6 and signed by the provided CSMS Root certificate> certificateType ChargingStationCertificate
16. The Charging Station responds with a CertificateSignedResponse	15. The Test System sends a CertificateSignedRequest With certificateChain <Certificate generated from the received CSR from step 9 and signed by the provided CSMS Root certificate> certificateType ChargingStationCertificate
Tool validations	
<p>* Step 2: Message: TriggerMessageResponse - status must be Accepted</p> <p>* 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.></p> <p>* Step 5: - The Charging Station shall not resend the SignCertificateRequest before the <Configured CertSigningWaitMinimum> expired</p> <p>* Step 8: - The Charging Station shall not resend the SignCertificateRequest before the <Configured CertSigningWaitMinimum> times 2 expired</p> <p>* Step 12, 14, 16: Message: CertificateSignedResponse - status must be Accepted or Rejected</p>	
<p>Post scenario validations: Note: It does not matter whether Charging Station accepts first or last or all certificates. At least one CertificateSignedResponse must have status Accepted</p>	

TC_A_19_CS: Upgrade Charging Station Security Profile - Accepted

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	<p>1. The Test System sends a SetNetworkProfileRequest with configurationSlot is <Configured configurationSlot> or <Configured configurationSlot2> depending on which one is already in use</p> <ul style="list-style-type: none"> - connectionData.messageTimeout <Configured messageTimeout> - connectionData.ocppCsmsUrl <Configured ocppCsmsUrl> - connectionData.ocppInterface <Configured ocppInterface> - connectionData.ocppVersion OCPP20 - connectionData.securityProfile <Configured securityProfile + 1> <p>Note(s):</p> <ul style="list-style-type: none"> - The Test System checks the ActiveNetworkProfile variable to see which slot is currently active. - The Test System prevents overwriting the NetworkProfile at the active slot, as this is not recommended.
4. The Charging Station responds with a SetVariablesResponse	<p>3. The Test System sends a SetVariablesRequest with variable.name is "NetworkConfigurationPriority"</p> <p>component.name is "OCPPCommCtrlr"</p> <p>attributeValue is <configurationSlot set at Step 1, the other configured configurationSlot></p>
6. The Charging Station responds with a ResetResponse	<p>5. The Test System sends a ResetRequest with type OnIdle</p>
	<p>7. The Test System restarts the WebSocket server using <Configured securityProfile + 1></p>
8. The Charging Station reconnects to the Test System using <Configured securityProfile + 1>	<p>9. The Test System accepts the connection attempt.</p>
10. Execute Reusable State Booted	
12. The Charging Station responds with GetVariablesResponse	<p>11. Test System sends GetVariablesRequest with:</p> <ul style="list-style-type: none"> - variable.name = "SecurityProfile" - component.name = "SecurityCtrlr"

Main (Test scenario)	
14. The Charging Station responds with GetVariablesResponse	13. Test System sends GetVariablesRequest with: - variable.name = "NetworkConfigurationPriority" - component.name = "OCPPCommCtrlr"
<i>The following steps are only executed when this testcase is upgrading from Security Profile 1 to Security Profile 2.</i>	
16. The Charging Station does NOT reconnect to the Test System using Security Profile 1.	15. The Test System closes the connection and restarts the WebSocket server using Security profile 1 and waits the <Configured long operation timeout>.
18. The Charging Station reconnects to the Test System using Security Profile 2.	17. The Test System restarts the WebSocket server using Security Profile 2.

Tool validations
<p>* Step 2: Message SetNetworkProfileResponse - status <i>Accepted</i></p> <p>* Step 4: Message SetVariablesResponse - setVariableResult[0].attributeStatus <i>Accepted OR RebootRequired</i></p> <p>* Step 6: Message ResetResponse - status <i>Accepted</i></p> <p>* Step 12: Message GetVariablesResponse - getVariableResult[0].attributeValue <i><Configured securityProfile + 1></i></p> <p>* Step 14: Message GetVariablesResponse - getVariableResult[0].attributeValue Does not contain the configurationSlot with the previous (lower) security profile</p> <p>Post scenario validations: - N/a</p>

TC_A_20_CS: Upgrade Charging Station Security Profile - No valid CSMSRootCertificate installed

Test case name	Upgrade Charging Station Security Profile - No valid CSMSRootCertificate installed
Test case Id	TC_A_20_CS
Use case Id(s)	A05
Requirement(s)	A05.FR.02
System under test	Charging Station
Description	The CSMS is able to change the connectionData at the Charging Station. By doing this it is able to upgrade the connection to a higher security profile.
Purpose	To verify if the Charging Station is able to reject upgrading to a higher security profile when it does not have a valid CSMSRootCertificate installed.
Prerequisite(s)	<ul style="list-style-type: none"> - The Test System connectionData configuration for SUT Charging Station only allows for ip addresses the Test System is able to bind. - The Charging Station supports at least 2 security profiles, one of which is security profile 1. - The Charging Station does not have a valid CSMSRootCertificate installed. - The first Test System connectionData configuration slot must be configured for security profile 1. - The second Test System connectionData configuration slot must be configured for security profile 2 or 3. - The Charging Station is connected using security profile 1. - When starting this testcase the Test System will start another webSocket server for the second connectionData slot.

Before (Preparations)

Configuration State:
 OCPPCommCtrlr.NetworkConfigurationPriority is <ActiveNetworkProfile slot> (All others are removed)

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 Test System sends a SetNetworkProfileRequest with - configurationSlot is <Configured configurationSlot2> or <Configured configurationSlot> (the one currently not used for the active connection) - 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 Test System sends a SetVariablesRequest with variable.name is "NetworkConfigurationPriority" component.name is "OCPPCommCtrlr" attributeValue is <configurationSlot set at step 1>,<previous configurationSlot>

Tool validations
<p>* Step 2:</p> <p>Message SetNetworkProfileResponse</p> <p>- status <i>Accepted</i> or <i>Rejected</i></p> <p>* Step 4:</p> <p>Message SetVariablesResponse</p> <p>- setVariableResult[0].attributeStatus <i>Rejected</i></p>
<p>Post scenario validations:</p> <p>- N/a</p>

TC_A_21_CS: Upgrade Charging Station Security Profile - No valid ChargingStationCertificate installed

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 security profile 3.
Purpose	To verify if the Charging Station is able to reject upgrading to a security profile 3 when it does not have a valid ChargingStationCertificate installed.
Prerequisite(s)	<ul style="list-style-type: none"> - The Test System connectionData configuration for SUT Charging Station only allows for ip addresses the Test System 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. - When starting this testcase the Test System 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 Test System 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 3
4. The Charging Station responds with a SetVariablesResponse	3. The Test System 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

Tool validations
Post scenario validations: - N/a

TC_A_22_CS: Upgrade Charging Station Security Profile - Downgrade security profile - Rejected

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 security profile 1.
Purpose	To verify if the Charging Station is able to reject downgrading to security profile 1.
Prerequisite(s)	<ul style="list-style-type: none"> - The Test System connectionData configuration for SUT Charging Station only allows for ip addresses the Test System is able to bind. - The Charging Station supports security profile 2 and/or 3. - The Charging Station has a connection using security profile 2 or 3. - When starting this testcase the Test System will start another webSocket server for the second connectionData slot.

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

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetNetworkProfileResponse	<p>1. The Test System sends a SetNetworkProfileRequest with:</p> <ul style="list-style-type: none">- configurationSlot is <i><Configured configurationSlot></i> or <i><Configured configurationSlot2></i> depending on which one is already in use- connectionData.messageTimeout <i><Configured messageTimeout2></i>- connectionData.ocppCsmsUrl <i><ocppCsmsUrl that is not currently active></i>- connectionData.ocppInterface <i><Configured ocppInterface2></i>- connectionData.ocppVersion <i>OCPP20</i>- connectionData.securityProfile <i>1</i>

Tool validations
* Step 2: Message SetNetworkProfileResponse - status <i>Rejected</i>
Post scenario validations: - N/a

B Provisioning**TC_B_01_CS: Cold Boot Charging Station - Accepted**

Test case name	Cold Boot Charging Station - Accepted
Test case Id	TC_B_01_CS
Use case Id(s)	B01
Requirement(s)	B01.FR.01, B01.FR.05, B01.FR.09
System under test	Charging Station
Description	The booting mechanism allows a Charging Station to provide some general information about the Charging Station to the CSMS on startup AND it allows the Charging Station to request whether it is allowed to start sending other OCPP messages.
Purpose	To verify whether the Charging Station is able to perform the booting mechanism as described at the OCPP specification.
Prerequisite(s)	N/a

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

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>Booted</i>	

Tool validations
N/a
Post scenario validations: - A message to report the state of a connector has been received for all connectors.

TC_B_02_CS: Cold Boot Charging Station - Pending

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 Test System responds with a BootNotificationResponse with status <i>Pending</i> interval <i><Configured heartbeatInterval></i>
4. The Charging Station responds with SetVariablesResponse	3. Test System sends SetVariablesRequest with: - variable.name = "OfflineThreshold" - component.name = "OCPPCommCtrlr" - attributeValue = "300" - attributeType is omitted
6. The Charging Station responds with GetVariablesResponse	5. Test System sends GetVariablesRequest with: - variable.name = "OfflineThreshold" - component.name = "OCPPCommCtrlr" - attributeType is omitted
8. Charging Station responds with: GetBaseReportResponse	7. Test System sends GetBaseReportRequest with: - requestId = <i><Generated requestId></i> - reportBase = <i>FullInventory</i>
Charging Station	CSMS
9. Charging Station responds with: NotifyReportRequest	10. Test System sends NotifyReportResponse
<u>Note(s)</u> : - This step is repeated as often as needed to report all configuration variables.	

Main (Test scenario)	
12. The Charging Station responds with a RequestStartTransactionResponse	11. The Test System sends a RequestStartTransactionRequest <u>Note(s):</u> - This step is executed after the Test System received all <i>NotifyReport</i> messages. This is indicated by the <i>tbc</i> and <i>seqNo</i> fields.
14. The Charging Station responds with a TriggerMessageResponse	13. The Test System sends a TriggerMessageRequest with requestedMessage <i>BootNotification</i>
15. The Charging Station sends a BootNotificationRequest <u>Note(s):</u> - The Charging Station resends the <i>BootNotificationRequest</i> after having responded to the <i>TriggerMessageRequest</i> , so before the interval from the <i>BootNotificationResponse</i> has been passed.	16. The Test System responds with a BootNotificationResponse with status <i>Accepted</i> interval <i><Configured heartbeatInterval></i>
17. The Charging Station notifies the CSMS about the current state of all connectors.	18. The Test System responds accordingly.

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

TC_B_03_CS: Cold Boot Charging Station - Rejected

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 Test System responds with a BootNotificationResponse with status Rejected interval <Configured heartbeatInterval>
3. The Charging Station sends a BootNotificationRequest <u>Note(s):</u> - The Charging Station resends the BootNotificationRequest after x seconds, whereby x is equal to or greater than the interval from the BootNotificationResponse. - The Charging Station is not allowed to send any OCPP message in the meantime. - The Charging Station is allowed to close the connection until it needs to resend the BootNotificationRequest.	4. The Test System 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 Test System responds accordingly.

Tool validations
<p>* Step 5:</p> <p>Message: StatusNotificationRequest</p> <ul style="list-style-type: none"> - connectorStatus Available <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none"> - eventData[0].trigger Delta - eventData[0].actualValue "Available" - eventData[0].component.name "Connector" - eventData[0].variable.name "AvailabilityState"
<p>Post scenario validations:</p> <ul style="list-style-type: none"> - A message to report the state of a connector has been received for all connectors.

TC_B_30_CS: Cold Boot Charging Station - Pending/Rejected - SecurityError

Test case name	Cold Boot Charging Station - Pending/Rejected - SecurityError
Test case Id	TC_B_30_CS
Use case Id(s)	B03
Requirement(s)	B03.FR.08
System under test	Charging Station
Description	The booting mechanism allows a Charging Station to provide some general information about the Charging Station to the CSMS on startup AND it allows the Charging Station to request whether it is allowed to start sending other OCPP messages. The CSMS may respond to the BootNotificationRequest with status <i>Rejected</i> . During this state, the Charging Station is not allowed to send RPC Framework: CALL message that is NOT a BootNotificationRequest.
Purpose	To verify whether the Charging Station is able to handle unauthorized messages from the CSMS by responding with a SecurityError.
Prerequisite(s)	

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 Test System responds with a BootNotificationResponse with status <i>Rejected</i>
4. The Charging Station responds with RPC Framework: CALLERROR: SecurityError.	3. The Test System sends a GetBaseReportRequest with reportBase <i>FullInventory</i> <u>Note(s)</u> : The Test System will only send this request if the Charging Station does not disconnect

Tool validations
N/a
N/a

TC_B_06_CS: Get Variables - single value

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. Test System 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

TC_B_07_CS: Get Variables - multiple values

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 <i>300</i> AuthCtrlr.LocalAuthorizeOffline is <i>true</i>
Memory State: N/a
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
2. Charging Station responds with GetVariablesResponse with attributeStatus = <i>Accepted</i> .	1. Test System sends GetVariablesRequest with: - getVariableData[0].variable.name = <i>"OfflineThreshold"</i> - getVariableData[0].component.name = <i>"OCPPCommCtrlr"</i> - getVariableData[0].attributeType = <i>Actual</i> - getVariableData[1].variable.name = <i>"LocalAuthorizeOffline"</i> - getVariableData[1].component.name = <i>"AuthCtrlr"</i> - getVariableData[1].attributeType = <i>Actual</i>

Tool validations
<p>* Step 2:</p> <p>Message: GetVariablesResponse has (in arbitrary order)</p> <p>GetVariableResultType[0]:</p> <ul style="list-style-type: none"> - attributeStatus = <i>Accepted</i> - attributeType = <i>Actual</i> - attributeValue = <i>300</i> - component.name = <i>"OCPPCommCtrlr"</i> - variable.name = <i>"OfflineThreshold"</i> - attributeStatusInfo is absent or attributeStatusInfo.reasonCode = <i>"NoError"</i> <p>GetVariableResultType[1]:</p> <ul style="list-style-type: none"> - attributeStatus = <i>Accepted</i> - attributeType = <i>Actual</i> - attributeValue = <i>"true"</i> - component.name = <i>"AuthCtrlr"</i> - variable.name = <i>"LocalAuthorizeOffline"</i> - attributeStatusInfo is absent or attributeStatusInfo.reasonCode = <i>"NoError"</i> <p>Post scenario validations: N/A</p>

TC_B_32_CS: Get Variables - Unknown component

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. Test System sends GetVariablesRequest with: - variable.name = "OfflineThreshold" - component.name = "UnknownComponent" - attributeType is omitted

Tool validations
* Step 2: Message: GetVariablesResponse - getVariableResult[0].attributeStatus = <i>UnknownComponent</i> - getVariableResult[0].component.name = "UnknownComponent" - getVariableResult[0].variable.name = "OfflineThreshold"
Post scenario validations: N/A

TC_B_33_CS: Get Variables - Unknown variable

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. Test System 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

TC_B_34_CS: Get Variables - Not supported attribute type

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. Test System sends GetVariablesRequest with: - variable.name = "OfflineThreshold" - component.name = "OCPPCommCtrlr" - attributeType = Target

Tool validations
* Step 2: Message: GetVariablesResponse - getVariableResult[0].attributeStatus = <i>NotSupportedAttributeType</i> - getVariableResult[0].component.name = "OCPPCommCtrlr" - getVariableResult[0].variable.name = "OfflineThreshold" - getVariableResult[0].attributeType = Target
Post scenario validations: N/A

TC_B_09_CS: Set Variables - single value

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 = <i>Accepted</i> .	1. Test System sends SetVariablesRequest with: - variable.name = <i>"OfflineThreshold"</i> - component.name = <i>"OCPPCommCtrlr"</i> - attributeValue = <i>"300"</i> - attributeType <i>Actual</i>

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

TC_B_10_CS: Set Variables - multiple values

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 = <i>Accepted</i> .	<p>1. Test System sends SetVariablesRequest with:</p> <ul style="list-style-type: none">- setVariableData[0].variable.name = "OfflineThreshold"- setVariableData[0].component.name = "OCPPCommCtrlr"- setVariableData[0].attributeValue = "300"- setVariableData[0].attributeType = Actual -setVariableData[1].variable.name = "LocalAuthorizeOffline"- setVariableData[1].component.name = "AuthCtrlr"- setVariableData[1].attributeValue = "true"- setVariableData[0].attributeType = Actual

Tool validations
<p>* Step 2:</p> <p>Message: SetVariablesResponse has (in arbitrary order)</p> <p>SetVariableResultType[1]:</p> <ul style="list-style-type: none"> - attributeStatus = <i>Accepted</i> - attributeType = <i>Actual</i> - component.name = "OCPPCommCtrlr" - variable.name = "OfflineThreshold" - attributeStatusInfo is absent or attributeStatusInfo.reasonCode = "NoError" <p>SetVariableResultType[2]:</p> <ul style="list-style-type: none"> - attributeStatus = <i>Accepted</i> - attributeType = <i>Actual</i> - component.name = "AuthCtrlr" - variable.name = "LocalAuthorizeOffline" - attributeStatusInfo is absent or attributeStatusInfo.reasonCode = "NoError" <p>Post scenario validations:</p> <p>N/A</p>

TC_B_35_CS: Set Variables - Unknown component

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. Test System sends SetVariablesRequest with: - variable.name = "OfflineThreshold" - component.name = "UnknownComponent" - attributeType is omitted

Tool validations
* Step 2: Message: SetVariablesResponse - setVariableResult[0].attributeStatus = <i>UnknownComponent</i> - setVariableResult[0].component.name = "UnknownComponent" - setVariableResult[0].variable.name = "OfflineThreshold"
Post scenario validations: N/A

TC_B_36_CS: Set Variables - Unknown variable

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. Test System 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

TC_B_37_CS: Set Variables - Not supported attribute type

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. Test System 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

TC_B_11_CS: Set Variables - invalidly formatted values

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

Main (Test scenario)	
Charging Station	CSMS
<u>Notes:</u> 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. Test System sends SetVariablesRequest with - variable.name = "OfflineThreshold" - component.name = "OCPPCommCtrlr" - attributeValue = value
<u>Notes:</u> 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. Test System sends SetVariablesRequest with - variable.name = "LimitChangeSignificance" - component.name = "SmartChargingCtrlr" - attributeValue = value
<u>Notes:</u> Steps 5 and 6 will only be executed if this component/variable combination is readwrite	
6. Charging Station responds with SetVariablesResponse with If value not supported: attributeStatus = Rejected attributeStatusInfo = InvalidValue If component/variable/value supported: attributeStatus = Accepted	5. Test System sends SetVariablesRequest with: - variable.name = "AuthorizeRemoteStart" - component.name = "AuthCtrlr" - attributeValue = value
<u>Notes:</u> Steps 7 and 8 will only be executed if the CS supports this component/variable combination	

Main (Test scenario)	
<p>8. Charging Station responds with SetVariablesResponse with</p> <p><i>If value not supported:</i></p> <p>attributeStatus = <i>Rejected</i></p> <p>attributeStatusInfo = <i>InvalidValue</i></p> <p><i>If component/variable/value supported:</i></p> <p>attributeStatus = <i>Accepted</i></p>	<p>7. Test System sends SetVariablesRequest with:</p> <ul style="list-style-type: none"> - variable.name = <i>"NextTimeOffsetTransitionDateTime"</i> - component.name = <i>"ClockCtrlr"</i> - attributeValue = <i>value</i>
Tool validations	
<p>* Step 2:</p> <p>Message: SetVariablesResponse has</p> <p>SetVariableResultType</p> <ul style="list-style-type: none"> - attributeStatus = <i>Rejected/Accepted</i> - attributeType = <i>Actual</i> - component.name = <i>"OCPPCommCtrlr"</i> - variable.name = <i>"OfflineThreshold"</i> - attributeStatusInfo is absent or attributeStatusInfo.reasonCode = <i>InvalidValue</i> (not required) 	
<p>* Step 4:</p> <p>Message: SetVariablesResponse has</p> <p>SetVariableResultType</p> <ul style="list-style-type: none"> - attributeStatus = <i>Rejected/Accepted</i> - attributeType = <i>Actual</i> - component.name = <i>"AuthCtrlr"</i> - variable.name = <i>"AuthorizeRemoteStart"</i> - attributeStatusInfo is absent or attributeStatusInfo.reasonCode = <i>InvalidValue</i> (not required) 	
<p>* Step 6:</p> <p>Message: SetVariablesResponse has</p> <p>SetVariableResultType</p> <ul style="list-style-type: none"> - attributeStatus = <i>Rejected/Accepted</i> - attributeType = <i>Actual</i> - component.name = <i>"SmartChargingCtrlr"</i> - variable.name = <i>"LimitChangeSignificance"</i> - attributeStatusInfo is absent or attributeStatusInfo.reasonCode = <i>InvalidValue</i> (not required) 	
<p>* Step 8:</p> <p>Message: SetVariablesResponse has</p> <p>SetVariableResultType</p> <ul style="list-style-type: none"> - attributeStatus = <i>Rejected/Accepted</i> - attributeType = <i>Actual</i> - component.name = <i>"ClockCtrlr"</i> - variable.name = <i>"NextTimeOffsetTransitionDateTime"</i> - attributeStatusInfo is absent or attributeStatusInfo.reasonCode = <i>InvalidValue</i> (not required) 	
<p>Post scenario validations:</p> <p>N/A</p>	

TC_B_39_CS: Set Variables - Read-only

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. Test System sends SetVariablesRequest with: - variable.name = "MessageTimeout" - variable.instance = "Default" - component.name = "OCPPCommCtrlr" - attributeType is omitted

Tool validations
* Step 2: Message: SetVariablesResponse - setVariableResult[0].attributeStatus = <i>Rejected</i> - setVariableResult[0].component.name = "OCPPCommCtrlr" - setVariableResult[0].variable.name = "MessageTimeout" - setVariableResult[0].variable.instance = "Default"
Post scenario validations: N/A

TC_B_12_CS: Get Base Report - ConfigurationInventory

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. Test System sends GetBaseReportRequest with: - requestId = <Generated requestId> - reportBase = <i>ConfigurationInventory</i>
3. Charging Station responds with: NotifyReportRequest	4. Test System sends NotifyReportResponse
Step 3 and 4 are repeated as often as needed to report all configuration variables.	

Tool validations	
* Step 2: Message: GetBaseReportResponse - status = <i>Accepted</i> - statusInfo is absent or statusInfo.reasonCode = <i>"NoError"</i>	
* Step 3: Message: NotifyReportRequest - requestId = <Generated requestId> - generatedAt = <timestamp at charging station> - seqNo = 0 - if variableCharacteristics.dataType = <i>OptionList, SequenceList</i> or <i>MemberList</i> then valuesList must be provided.	
while tbc = <i>true</i>	Expect NotifyReportRequest - seqNo is incremented by 1
Post scenario validations: Check for all received variables: - variableCharacteristics are present - mutability = <i>ReadWrite</i> or <i>WriteOnly</i> 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.	

TC_B_13_CS: Get Base Report - FullInventory

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. Test System sends GetBaseReportRequest with: - requestId = <Generated requestId> - reportBase = FullInventory
3. Charging Station responds with: NotifyReportRequest	4. Test System sends NotifyReportResponse
Step 3 and 4 are repeated as often as needed to report all configuration variables.	

Tool validations	
<p>* Step 2:</p> <p>Message: GetBaseReportResponse</p> <ul style="list-style-type: none">- status = <i>Accepted</i>- statusInfo is absent or statusInfo.reasonCode = <i>"NoError"</i>	
<p>* Step 3:</p> <p>Message: NotifyReportRequest</p> <ul style="list-style-type: none">- requestId = <i><Generated requestId></i>- generatedAt = <i><timestamp at charging station></i>- seqNo = 0 - if variableCharacteristics.dataType = <i>OptionList, SequenceList or MemberList</i> then valuesList must be provided.	
while tbc = <i>true</i>	Expect NotifyReportRequest <ul style="list-style-type: none">- seqNo is incremented by 1
<p>Post scenario validations:</p> <p>Check for all received variables:</p> <ul style="list-style-type: none">- variableCharacteristics are present <p>Validate that as a minimum the required variables mentioned in section "Charging Infrastructure Related" are reported as well as the required variables in section "Referenced Components and Variables", that are relevant to each functional block that has been implemented.</p>	

TC_B_14_CS: Get Base Report - SummaryInventory

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. Test System sends GetBaseReportRequest with: - requestId = <Generated requestId> - reportBase = <i>SummaryInventory</i>
3. Charging Station responds with: NotifyReportRequest	4. Test System sends NotifyReportResponse
Step 3 and 4 are repeated as often as needed to report all configuration variables.	

Tool validations	
* Step 2: Message: GetBaseReportResponse - status = <i>Accepted</i> - statusInfo is absent or statusInfo.reasonCode = <i>"NoError"</i>	
* Step 3: Message: NotifyReportRequest - requestId = <Generated requestId> - generatedAt = <timestamp at charging station> - seqNo = 0	
while tbc = <i>true</i>	Expect NotifyReportRequest - seqNo is incremented by 1
Post scenario validations: Check for all received variables: - variableCharacteristics are present - if variableCharacteristics.dataType = <i>OptionList, SequenceList</i> or <i>MemberList</i> 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.	

TC_B_15_CS: Get Base Report - Not Supported base report

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. Test System sends GetBaseReportRequest with: - requestId = <Generated requestId> - reportBase = SummaryInventory
Note(s): - Test System 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

TC_B_16_CS: Get Custom Report - with component criteria

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, B08.FR.10, B08.FR.12, B08.FR.13, B08.FR.14, B06.FR.09
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
	1. Test System sends GetReportRequest with: - requestId = <Generated requestId> - componentCriteria = { Enabled }
2. Charging Station responds: GetReportResponse	
3. Charging Station sends NotifyReportRequest with: - reportData for all components that have the variable "Enabled" set to "true".	4. Test System responds with NotifyReportResponse
Steps 3 and 4 may be repeated multiple times until everything has been reported.	
Test System sends a <i>GetVariables</i> for all variables to match results.	
6. Charging Station responds with: GetVariablesResponse	5. Test System sends GetVariablesRequest

Tool validations	
<p>* Step 2: +2. Message: GetReportResponse with:</p> <ul style="list-style-type: none">- status = <i>Accepted</i>- statusInfo is absent or statusInfo = <i>"NoError"</i>	
<p>* step 3:</p> <p>Message: NotifyReportRequest with:</p> <ul style="list-style-type: none">- requestId = <i><Generated requestId></i>- generatedAt = <i><time of generation at charging station></i>- seqNo = <i>0</i>- reportData.variableCharacteristics are present- reportData.variable.name is <i>"Enabled"</i>- reportData.variableAttribute.value = <i>"true"</i> <p><i>Note: for Enabled there will only be an Actual value.</i></p> <p><i>It does not make any sense to have a MinSet, MaxSet or Target value for them.</i></p>	
While tbc = <i>true</i>	<p>Message: NotifyReportRequest</p> <ul style="list-style-type: none">- seqNo is incremented by one- reportData.variableCharacteristics are present- reportData.variable.name is <i>"Enabled"</i>- reportData.variableAttribute.value = <i>"true"</i>

Tool validations
<p>* Step 6:</p> <p>For component variables where NotifyReportRequest.reportData.variableAttribute.mutability from step 3 is not <i>WriteOnly</i></p> <ul style="list-style-type: none">- attributeStatus = <i>Accepted</i>- attributeValue = <i>true</i> <p>For component variables where NotifyReportRequest.reportData.variableAttribute.mutability from step 3 is <i>WriteOnly</i></p> <ul style="list-style-type: none">- attributeStatus = <i>Rejected</i>- attributeValue = <i><omitted></i>
<p>Post scenario validations:</p> <p>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 = { <i>Enabled</i> }.</p>

TC_B_17_CS: Get Custom Report - with component/variable

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. Test System 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. Test System responds with NotifyReportResponse

Tool validations
* Step 2: Message: GetReportResponse with: - status = <i>Accepted</i> - statusInfo is absent or statusInfo = "NoError"
* step 3: Message: NotifyReportRequest with: - requestId = <i>GetReportRequest requestId</i> - 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 = <i>OptionList</i> - reportData.variableCharacteristics.valuesList = "Available, Occupied, Reserved, Unavailable, Faulted" - reportData.variableAttribute.mutability = <i>ReadOnly</i> - reportData.variableAttribute.type = <i>Actual</i> Note: for AvailabilityState there will only be an <i>_Actual</i> value. It does not make any sense to have a <i>MinSet</i> , <i>MaxSet</i> or <i>Target</i> value for it.
Post scenario validations: N/A

TC_B_18_CS: Get Custom Report - with component criteria and component/variable

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 <i>Available</i> and with <i>Problem</i> .
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 <i>Available</i> with <i>AvailabilityState</i> ._	
2. Charging Station responds: GetReportResponse	1. Test System sends GetReportRequest with: - requestId = <Generated requestId> - componentCriteria = { <i>Available</i> } - componentVariable[0].component.name = "EVSE" - componentVariable[0].component.evse.id = 1 - componentVariable[0].variable.name = "AvailabilityState"
3. Charging Station sends NotifyReportRequest	4. Test System responds with NotifyReportResponse
Request EVSE::AvailabilityState from _Problem components. Note 1: Assuming EVSE #1 does not have <i>Problem</i> variable set._	
6. Charging Station responds: GetReportResponse	5. Test System sends GetReportRequest with: - requestId = <Generated requestId> - componentCriteria[0] = <i>Problem</i> - componentVariable[0].component.name = "EVSE" - componentVariable[0].component.evse.id = 1 - componentVariable[0].variable.name = "AvailabilityState"

Tool validations
* Step 2: Message: GetReportResponse with: - status = <i>Accepted</i> - statusInfo is absent or statusInfo = "NoError"

Tool validations
<p>* step 3:</p> <p>Message: NotifyReportRequest with:</p> <ul style="list-style-type: none"> - requestId = <i>GetReportRequest requestid</i> - 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 = <i>OptionList</i> - reportData.variableCharacteristics.valuesList = "Available, Occupied, Reserved, Unavailable, Faulted" - reportData.variableAttribute.mutability = <i>ReadOnly</i> - reportData.variableAttribute.type = <i>Actual</i> <p>Note: for <i>AvailabilityState</i> there will only be an <i>_Actual</i> value. It does not make any sense to have a <i>MinSet</i>, <i>MaxSet</i> or <i>Target</i> value for it.</p>
<p>* Step 6:</p> <p>Message: GetReportResponse with:</p> <ul style="list-style-type: none"> - status = <i>EmptyResultSet</i> - statusInfo is absent or statusInfo.reasonCode = "NotFound"
<p>Post scenario validations:</p> <p>N/A</p>

TC_B_19_CS: Get Custom Report - for unknown component criteria

NOTE

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 <i>NotSupported</i> 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. Test System sends GetReportRequest with: - requestId = <i><Generated requestId></i> - componentCriteria = { <i>Available</i> , <i><Configured Unsupported componentCriteria></i> } - *componentVariable is absent

Tool validations
* Step 2 Message: GetReportResponse - status = <i>NotSupported</i> - statusInfo is absent or statusInfo.reasonCode = " <i>UnsupportedParam</i> " or statusInfo.reasonCode = " <i>InvalidValue</i> "
Post scenario validations: N/A

TC_B_20_CS: Reset Charging Station - Without ongoing transaction - OnIdle

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 Test System sends a ResetRequest with type OnIdle
<u>Note(s):</u> - Charging Station reboots	
3. The Charging Station sends a BootNotificationRequest	4. The Test System responds with a BootNotificationResponse with status Accepted
5. The Charging Station notifies the CSMS about the current state of all connectors.	6. The Test System 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.

TC_B_21_CS: Reset Charging Station - With Ongoing Transaction - OnIdle

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	<p>This test case covers how the CSMS can remotely request the Charging Station to reset itself by sending a ResetRequest during a transaction. When ResetRequest "OnIdle" is send the charging stations schedules a reboot after all transactions are stopped.</p> <p>This could for example be necessary if the Charging Station is not functioning correctly.</p>
Purpose	To verify if the Charging Station is able to perform the reset mechanism while there is an ongoing transaction as described at the OCPP specification.
Prerequisite(s)	n/a

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a ResetResponse	1. The Test System sends a ResetRequest with type OnIdle
3. Execute Reusable State <i>StopAuthorized</i>	
4. Execute Reusable State <i>EVConnectedPostSession</i>	
5. Execute Reusable State <i>EVDisconnected</i>	
<u>Notes(s)</u> : Steps 4 and 5 will only be executed if TxStartPoint does not contain: EnergyTransferStarted, DataSigned, PowerPathClosed, or Authorized	
6. Execute Reusable State <i>ParkingBayUnoccupied</i>	
<u>Notes(s)</u> : 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 Test System responds with a BootNotificationResponse
9. The Charging Station notifies the CSMS about the current state of all connectors.	10. The Test System responds accordingly.
11. The Charging Station sends a SecurityEventNotificationRequest	12. The Test System responds with a SecurityEventNotificationResponse

Tool validations
<p>* Step 2: Message ResetResponse - status <i>Scheduled</i></p> <p>* Step 7: Message BootNotificationRequest - reason <i>ScheduledReset</i></p> <p>* Step 9: Message: StatusNotificationRequest - If the transaction was stopped at step 3, then — for the connector involved in the transaction: connectorStatus <i>Occupied</i> — for other connectors of this EVSE: connectorStatus <i>Available</i> or <i>Unavailable</i> Else connectorStatus <i>Available</i> Message: NotifyEventRequest - eventData[0].trigger <i>Delta</i> - eventData[0].component.name <i>"Connector"</i> - eventData[0].variable.name <i>"AvailabilityState"</i> - If the transaction was stopped at step 3, then — for the connector involved in the transaction: eventData[0].actualValue <i>Occupied</i> — for other connectors of this EVSE: eventData[0].actualValue <i>Available</i> or <i>Unavailable</i> - Else eventData[0].actualValue <i>"Available"</i></p> <p>* Step 11: Message: SecurityEventNotificationRequest - type <i>StartupOfTheDevice</i> or <i>ResetOrReboot</i></p>
<p>Post scenario validations: - A message to report the state of a connector has been received for all connectors.</p>

TC_B_22_CS: Reset Charging Station - With Ongoing Transaction - Immediate

Test case name	Reset Charging Station - With Ongoing Transaction - Immediate
Test case Id	TC_B_22_CS
Use case Id(s)	B12
Requirement(s)	B12.FR.02, B12.FR.04, E07.FR.03, B01.FR.03
System under test	Charging Station
Description	<p>This test case covers how the CSMS can remotely request the Charging Station to reset itself by sending a ResetRequest during a transaction. When ResetRequest "Immediate" is send the charging stations will try to stop all transactions before rebooting.</p> <p>This could for example be necessary if the Charging Station is not functioning correctly.</p>
Purpose	To verify if the Charging Station is able to perform the reset mechanism as described at the OCPP specification.
Prerequisite(s)	n/a

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a ResetResponse	1. The Test System sends a ResetRequest with type Immediate
3. The Charging Station sends a TransactionEventRequest	4. The Test System responds with a TransactionEventResponse
<u>Note(s):</u> - Charging Station reboots	
5. The Charging Station sends a BootNotificationRequest	6. The Test System responds with a BootNotificationResponse with status Accepted
7. The Charging Station notifies the CSMS about the current state of all connectors.	8. The Test System responds accordingly.

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**

- for the connector involved in the transaction: **connectorStatus** *Occupied*

- for other connectors of the same EVSE: **connectorStatus** *Available* or *Unavailable*

Message: **NotifyEventRequest**

- **eventData[0].trigger** *Delta*

- for the connector involved in the transaction: **eventData[0].actualValue** *Occupied*

- for other connectors of the same EVSE: **eventData[0].actualValue** *Available* or *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.

TC_B_23_CS: Reset Charging Station - Unavailable persists reset

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: <i>Unavailable</i> for <Configured connectorId>
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a ResetResponse	1. The Test System sends a ResetRequest with type OnIdle
Note(s): - The Charging Station reboots	
3. The Charging Station sends a BootNotificationRequest	4. The Test System responds with a BootNotificationResponse with status Accepted
5. The Charging Station notifies the CSMS about the current state of all connectors.	6. The Test System responds accordingly.

Tool validations
<p>* Step 2:</p> <p>Message ResetResponse</p> <ul style="list-style-type: none">- status <i>Accepted</i> <p>* Step 3:</p> <p>Message BootNotificationRequest</p> <p>reason <i>RemoteReset</i></p> <p>* Step 5:</p> <p>For <Configured connectorId>:</p> <p>Message: StatusNotificationRequest</p> <ul style="list-style-type: none">- connectorStatus <i>Unavailable</i> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none">- eventData[0].trigger <i>Delta</i>- eventData[0].actualValue <i>"Unavailable"</i>- eventData[0].component.name <i>"Connector"</i>- eventData[0].variable.name <i>"AvailabilityState"</i> <p>For <Other connector(s)>:</p> <p>Message: StatusNotificationRequest</p> <ul style="list-style-type: none">- connectorStatus <i>Available</i> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none">- eventData[0].trigger <i>Delta</i>- eventData[0].actualValue <i>"Available"</i>- eventData[0].component.name <i>"Connector"</i>- eventData[0].variable.name <i>"AvailabilityState"</i>
<p>Post scenario validations:</p> <ul style="list-style-type: none">- A message to report the state of a connector has been received for all connectors.

TC_B_24_CS: Reset Charging Station - Reserved persists reset

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: <i>Reserved</i> for <Configured connectorId>
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a ResetResponse	1. The Test System sends a ResetRequest with type Immediate
<u>Note(s):</u> - The Charging Station reboots	
3. The Charging Station sends a BootNotificationRequest	4. The Test System responds with a BootNotificationResponse with status Accepted
5. The Charging Station notifies the CSMS about the current state of all connectors.	6. The Test System responds accordingly.

Tool validations
<p>* Step 2:</p> <p>Message ResetResponse</p> <ul style="list-style-type: none">- status <i>Accepted</i> <p>* Step 3:</p> <p>Message BootNotificationRequest</p> <p>reason <i>RemoteReset</i></p> <p>* Step 5:</p> <p>For <Configured connectorId>:</p> <p>Message: StatusNotificationRequest</p> <ul style="list-style-type: none">- connectorStatus <i>Reserved</i> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none">- eventData[0].trigger <i>Delta</i>- eventData[0].actualValue <i>"Reserved"</i>- eventData[0].component.name <i>"Connector"</i>- eventData[0].variable.name <i>"AvailabilityState"</i> <p>For <Other connector(s)>:</p> <p>Message: StatusNotificationRequest</p> <ul style="list-style-type: none">- connectorStatus <i>Available</i> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none">- eventData[0].trigger <i>Delta</i>- eventData[0].actualValue <i>"Available"</i>- eventData[0].component.name <i>"Connector"</i>- eventData[0].variable.name <i>"AvailabilityState"</i>
<p>Post scenario validations:</p> <ul style="list-style-type: none">- A message to report the state of a connector has been received for all connectors.

TC_B_41_CS: Reset Charging Station - With multiple ongoing transactions - OnIdle

Test case name	Reset Charging Station - With multiple ongoing transactions - OnIdle
Test case Id	TC_B_41_CS
Use case Id(s)	B12
Requirement(s)	B12.FR.01, B12.FR.03, E07.FR.03
System under test	Charging Station
Description	<p>This test case covers how the CSMS can remotely request the Charging Station to reset itself by sending a ResetRequest during a transaction. When ResetRequest "OnIdle" is send the charging stations schedules a reboot after all transactions are stopped.</p> <p>This could for example be necessary if the Charging Station is not functioning correctly.</p>
Purpose	To verify if the Charging Station is able to perform the reset mechanism while there are multiple ongoing transactions as described at the OCPP specification.
Prerequisite(s)	The Charging Station has more than one EVSE.

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferStarted</i> for EVSE.id = 1 State is <i>EnergyTransferStarted</i> for EVSE.id = 2

Main (Test scenario)	
Charging Station	CSMS
	1. The Test System sends a ResetRequest with type <i>OnIdle</i>
2. The Charging Station responds with a ResetResponse	
3. Execute Reusable State <i>StopAuthorized</i> for EVSE.id = 1	
4. Execute Reusable State <i>EVConnectedPostSession</i> for EVSE.id = 1	
5. Execute Reusable State <i>EVDisconnected</i> for EVSE.id = 1	
6. Execute Reusable State <i>ParkingBayUnoccupied</i> for EVSE.id = 1	
7. Execute Reusable State <i>StopAuthorized</i> for EVSE.id = 2	
8. Execute Reusable State <i>EVConnectedPostSession</i> for EVSE.id = 2	
<u>Note(s):</u> If TxStopPoint contains one of the following values; Authorized, EnergyTransfer, PowerPathClosed, DataSigned. Then the transaction will have ended at the <i>EVConnectedPostSession</i> state AND the Charging Station will proceed with resetting itself. Proceed to step 11 Else proceed with step 9.	
9. Execute Reusable State <i>EVDisconnected</i> for EVSE.id = 2	
<u>Note(s):</u> If TxStopPoint contains the value EVConnected. Then the transaction will have ended at the <i>EVDisconnected</i> state AND the Charging Station will proceed with resetting itself. Proceed to step 11 Else proceed with step 10	
10. Execute Reusable State <i>ParkingBayUnoccupied</i> for EVSE.id = 2	
<u>Note(s):</u> 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 Test System responds with a BootNotificationResponse

Main (Test scenario)	
13. The Charging Station notifies the CSMS about the current state of all connectors.	14. The Test System responds accordingly.

Tool validations
<p>* Step 2: Message ResetResponse - status <i>Scheduled</i></p> <p>* Step 11: Message BootNotificationRequest - reason <i>ScheduledReset</i></p> <p>* Step 13: Message: StatusNotificationRequest - If the transaction was stopped at step 3, then – for the connector involved in the transaction: connectorStatus <i>Occupied</i> – for other connectors of this EVSE: connectorStatus <i>Available</i> or <i>Unavailable</i> - Else connectorStatus <i>Available</i></p> <p>Message: NotifyEventRequest - eventData[0].trigger <i>Delta</i> - eventData[0].component.name <i>"Connector"</i> - eventData[0].variable.name <i>"AvailabilityState"</i> - If the transaction was stopped at step 3, then – for the connector involved in the transaction: eventData[0].actualValue <i>Occupied</i> – for other connectors of this EVSE: eventData[0].actualValue <i>Available</i> or <i>Unavailable</i> - Else eventData[0].actualValue <i>"Available"</i></p>
<p>Post scenario validations: - A message to report the state of a connector has been received for all connectors.</p>

TC_B_25_CS: Reset EVSE - Without ongoing transaction

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 Test System sends a ResetRequest with type OnIdle and <i>*evseld*<Configured evseld></i>
<u>Note(s)</u> : - <i><Configured evseld> reboots</i>	

Tool validations
* Step 2: Message ResetResponse - status <i>Accepted</i>
Post scenario validations: - N/a

TC_B_26_CS: Reset EVSE - With Ongoing Transaction - OnIdle

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	<p>This test case covers how the CSMS can remotely request the Charging Station to reset an EVSE by sending a ResetRequest during a transaction. When ResetRequest "OnIdle" is send the charging stations schedules a reboot after all transactions are stopped.</p> <p>This could for example be necessary if the Charging Station is not functioning correctly.</p>
Purpose	To verify if the Charging Station is able to perform the reset mechanism as described at the OCPP specification.
Prerequisite(s)	Individual resetting EVSE supported

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a ResetResponse	1. The Test System sends a ResetRequest with type OnIdle and evseld <Configured evseld>
3. Execute Reusable State <i>StopAuthorized</i>	
4. Execute Reusable State <i>EVConnectedPostSession</i>	
5. Execute Reusable State <i>EVDisconnected</i>	
6. Execute Reusable State <i>ParkingBayUnoccupied</i>	

Tool validations
* Step 2: Message ResetResponse - status <i>Scheduled</i>
Post scenario validations: N/a

TC_B_27_CS: Reset EVSE - With Ongoing Transaction - Immediate

Test case name	Reset EVSE - With Ongoing Transaction - Immediate
Test case Id	TC_B_27_CS
Use case Id(s)	B12
Requirement(s)	B12.FR.02, B12.FR.08, E07.FR.03
System under test	Charging Station
Description	<p>This test case covers how the CSMS can remotely request the Charging Station to reset an EVSE by sending a ResetRequest during a transaction. When ResetRequest "Immediate" is send the charging stations will try to stop all transactions before rebooting.</p> <p>This could for example be necessary if the Charging Station is not functioning correctly.</p>
Purpose	To verify if the Charging Station is able to perform the reset mechanism as described at the OCPP specification.
Prerequisite(s)	Individual resetting EVSE supported

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a ResetResponse	1. The Test System sends a ResetRequest with type Immediate and <i>*evseld*<Configured evseld></i>
3. The Charging Station sends a TransactionEventRequest	4. The Test System responds with a TransactionEventResponse
<u>Note(s):</u> - The EVSE reboots	

Tool validations
* Step 2: Message ResetResponse - status <i>Accepted</i> * Step 3: Message TransactionEventRequest - eventType <i>Ended</i> - triggerReason <i>ResetCommand</i> - transactionInfo.chargingState <i>EVConnected</i> - transactionInfo.stoppedReason <i>ImmediateReset</i>
Post scenario validations: - N/a

TC_B_28_CS: Reset EVSE - Not Supported

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 Test System sends a ResetRequest with type OnIdle and *evseld*<Configured evseld>

Tool validations
* Step 2: Message ResetResponse - status <i>Rejected</i>
Post scenario validations: - N/a

TC_B_29_CS: Reset EVSE - With ongoing transaction - Not Supported

Test case name	Reset EVSE - With ongoing transaction - Not Supported
Test case Id	TC_B_29_CS
Use case Id(s)	B11
Requirement(s)	B12.FR.01, B12.FR.09
System under test	Charging Station
Description	This test case covers how the CSMS can request the Charging Station to reset an EVSE by sending a ResetRequest with ongoing transaction while it is not supported by the Charging Station.
Purpose	To verify if the Charging Station is able to perform the reset mechanism as described at the OCPP specification.
Prerequisite(s)	Charging Station does not support resetting individual EVSE

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a ResetResponse	1. The Test System sends a ResetRequest with type OnIdle and evseld <Configured evseld>

Tool validations
* Step 2: Message ResetResponse - status <i>Rejected</i>
Post scenario validations: - N/a

TC_B_43_CS: Set new NetworkConnectionProfile - Rejected

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	<p>1. The Test System sends a SetNetworkProfileRequest with:</p> <ul style="list-style-type: none"> - configurationSlot is 999 - connectionData.messageTimeout <Configured messageTimeout> - connectionData.ocppCsmsUrl <Configured ocppCsmsUrl> - connectionData.ocppInterface <Configured ocppInterface> - connectionData.ocppVersion OCPP20 - connectionData.securityProfile <Configured securityProfile>

Tool validations
<p>* Step 2:</p> <p>Message SetNetworkProfileResponse</p> <ul style="list-style-type: none"> - status <i>Rejected</i>
<p>Post scenario validations:</p> <ul style="list-style-type: none"> - N/a

TC_B_45_CS: Migrate to new ConnectionProfile - Success - Same CSMS Root

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	<p>1. The Test System sends a SetNetworkProfileRequest with configurationSlot is <Configured configurationSlot> or <Configured configurationSlot2> depending on which one is already in use</p> <ul style="list-style-type: none"> - 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> <p><u>Note(s):</u></p> <ul style="list-style-type: none"> - The Test System checks the ActiveNetworkProfile variable to see which slot is currently active. - The Test System prevents overwriting the NetworkProfile at the active slot, as this is not recommended.
4. The Charging Station responds with a SetVariablesResponse	<p>3. The Test System sends a SetVariablesRequest with variable.name is "NetworkConfigurationPriority"</p> <p>component.name is "OCPPCommCtrlr"</p> <p>attributeValue is <configurationSlot set at Step 1, the other configured configurationSlot></p>
6. The Charging Station responds with a ResetResponse	<p>5. The Test System sends a ResetRequest with type OnIdle</p>
<p>7. Execute Reusable State <i>Booted</i></p> <p><u>Note(s):</u></p> <ul style="list-style-type: none"> - The Charging Station connects to the slot configured at step 1. 	

Tool validations
<div>* Step 2: Message SetNetworkProfileResponse - status <i>Accepted</i></div> <div>* Step 4: Message SetVariablesResponse - setVariableResult[0].attributeStatus <i>Accepted OR RebootRequired</i></div> <div>* Step 6: Message ResetResponse - status <i>Accepted</i></div>
<div>Post scenario validations: - N/a</div>

TC_B_46_CS: Migrate to new ConnectionProfile - Fallback mechanism - Same CSMS Root

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	<p>1. The Test System sends a SetNetworkProfileRequest with configurationSlot is <i><Configured configurationSlot></i> or <i><Configured configurationSlot2></i> depending on which one is already in use</p> <ul style="list-style-type: none"> - connectionData.messageTimeout <i><Configured messageTimeout></i> or <i><Configured messageTimeout2></i> - connectionData.ocppCsmsUrl <i><ocppCsmsUrl that is not currently active></i> - connectionData.ocppInterface <i><Configured ocppInterface></i> or <i><Configured ocppInterface2></i> - connectionData.ocppVersion <i>OCPP20</i> - connectionData.securityProfile <i><Configured securityProfile></i> or <i><Configured securityProfile2></i> <p><u>Note(s):</u></p> <ul style="list-style-type: none"> - The Test System checks the <i>ActiveNetworkProfile</i> variable to see which slot is currently active. - The Test System prevents overwriting the <i>NetworkProfile</i> at the active slot, as this is not recommended.
4. The Charging Station responds with a SetVariablesResponse	<p>3. The Test System sends a SetVariablesRequest with variable.name is "NetworkConfigurationPriority"</p> <p>component.name is "OCPPCommCtrlr"</p> <p>attributeValue is <i><configurationSlot set at Step 1, the other configured configurationSlot></i></p>
6. The Charging Station responds with a ResetResponse	<p>5. The Test System sends a ResetRequest with type <i>OnIdle</i></p>
	<p>7. The Test System will NOT respond to the two connection request from the Charging Station from the first connectionSlot.</p>
	<p>8. The Test System will accept the connection request from the Charging Station from the second connectionSlot.</p>

Main (Test scenario)
<u>Note(s)</u> : Set the <Configured Long Operation Time Out> so that Steps 7 and 8 can be completed in this time period.
9. Execute Reusable State <i>Booted</i>
<u>Note(s)</u> : - The Charging Station connects to the second slot configured at the NetworkConfigurationPriority at step 3.
Tool validations
* Step 2: Message SetNetworkProfileResponse - status <i>Accepted</i>
* Step 4: Message SetVariablesResponse - setVariableResult[0].attributeStatus <i>Accepted OR RebootRequired</i>
* Step 6: Message ResetResponse - status <i>Accepted</i>
Post scenario validations: - N/a

TC_B_47_CS: Migrate to new ConnectionProfile - Fallback after NetworkProfileConnectionAttempts per NetworkConfigurationPriority failed - New CSMS Root - New CSMS

Test case name	Migrate to new ConnectionProfile - Fallback after NetworkProfileConnectionAttempts per NetworkConfigurationPriority failed - New CSMS Root - New CSMS
Test case Id	TC_B_47_CS
Use case Id(s)	B09,B10,M05
Requirement(s)	B10.FR.07,M05.FR.15,M05.FR.16
System under test	Charging Station
Description	The CSMS updates the connection details on the Charging Station. For instance in preparation of a migration to a new CSMS.
Purpose	To verify if the Charging Station is able to correctly handle the fallback mechanism in the case it fails to connect to the other CSMS.
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station supports AS-2: AdditionalRootCertificateCheck. - Configured (new) CSMS Root certificate 2 must be signed by the configured (old) CSMS Root certificate 2. - At least two configuration slots for networkConnectionProfiles must be supported

Before (Preparations)
Configuration State: OCPPCommCtrlr.NetworkProfileConnectionAttempts is 1
Memory State: <i>CertificateInstalled</i> 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	<p>1. The Test System sends a SetNetworkProfileRequest with configurationSlot is <Configured configurationSlot> or <Configured configurationSlot2> depending on which one is already in use</p> <ul style="list-style-type: none"> - 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> <p><u>Note(s):</u></p> <ul style="list-style-type: none"> - The Test System checks the ActiveNetworkProfile variable to see which slot is currently active. - The Test System prevents overwriting the NetworkProfile at the active slot, as this is not recommended.
4. The Charging Station responds with a SetVariablesResponse	<p>3. The Test System sends a SetVariablesRequest with variable.name is "NetworkConfigurationPriority" component.name is "OCPPCommCtrlr" attributeValue is <configurationSlot set at Step 1, the other configured configurationSlot></p>
6. The Charging Station responds with a ResetResponse	<p>5. The Test System sends a ResetRequest with type OnIdle</p>

Main (Test scenario)	
<p>8. During the TLS handshake the Charging Station validates the CSMS certificate.</p> <p><u>Note(s):</u> - This connection attempt fails, because the Charging Station will use the new CSMS Root certificate to validate the CSMS certificate.</p>	<p>7. During the TLS handshake the Test System provides a CSMS certificate which is signed by the <Configured old CSMS Root certificate></p>
<p>9. The Charging Station switches back to the previous networkprofile configuration and validates the CSMS certificate, using the (fallback) CSMS Root certificate.</p> <p><u>Note(s):</u> - This connection attempt succeeds, because the Charging Station will now use the (old) CSMS Root certificate to validate the CSMS certificate.</p>	
10. Execute Reusable State <i>Booted</i>	
<p>12. The Charging Station responds with a GetInstalledCertificateIdsResponse</p>	<p>11. The Test System sends a GetInstalledCertificateIdsRequest with certificateType is <i>CSMSRootCertificate</i></p>
Tool validations	
<p>* Step 6: Message ResetResponse - status <i>Accepted</i></p> <p>* Step 12: Message: GetInstalledCertificateIdsResponse - status must be <i>Accepted</i> - certificateHashDataChain must contain an entry with following values: - certificateType is <i>CSMSRootCertificate</i> - certificateHashData contains <HashData from configured old CSMS Root certificate></p>	
<p>Post scenario validations: - N/a</p>	

TC_B_49_CS: Migrate to new ConnectionProfile - Fallback after NetworkProfileConnectionAttempts per NetworkConfigurationPriority failed - Same CSMS Root

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.03, B10.FR.07
System under test	Charging Station
Description	The CSMS updates the connection details on the Charging Station. For instance in preparation of a migration to a new CSMS.
Purpose	To verify if the Charging Station is able to correctly handle the fallback mechanism in the case it fails to connect to the other CSMS.
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station supports C-47: mechanism implemented & Reconnect after NetworkProfileConnectionAttempts - At least two configuration slots for networkConnectionProfiles must be supported

Before (Preparations)
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

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetNetworkProfileResponse	<p>1. The Test System sends a SetNetworkProfileRequest with configurationSlot is <Configured configurationSlot> or <Configured configurationSlot2> depending on which one is already in use</p> <ul style="list-style-type: none"> - 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> <p><u>Note(s):</u></p> <ul style="list-style-type: none"> - The Test System checks the ActiveNetworkProfile variable to see which slot is currently active. - The Test System prevents overwriting the NetworkProfile at the active slot, as this is not recommended.
4. The Charging Station responds with a SetVariablesResponse	<p>3. The Test System sends a SetVariablesRequest with variable.name is "NetworkConfigurationPriority"</p> <p>component.name is "OCPPCommCtrlr"</p> <p>attributeValue is <configurationSlot set at Step 1, the other configured configurationSlot></p>

Main (Test scenario)	
6. The Charging Station responds with a ResetResponse	5. The Test System sends a ResetRequest with type OnIdle
7. The Charging Station tries to connect to the alternative internal Test System endpoint. <u>Note(s):</u> - 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 Test System.
9. The Charging Station switches back to the previous networkprofile configuration and reconnects to the Test System.	10. The connection attempt is not accepted by the Test System.
11. The Charging Station waits for at least the duration of the configured RetryBackOffWaitMinimum and reconnects to the Test System.	12. The connection attempt is accepted by the Test System.
13. Execute Reusable State <i>Booted</i>	

Tool validations
* Step 6: Message ResetResponse - status <i>Accepted</i>
Post scenario validations: - N/a

TC_B_50_CS: Migrate to new ConnectionProfile - Success - New CSMS Root - New CSMS

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: <i>CertificateInstalled</i> 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	<p>1. The Test System sends a SetNetworkProfileRequest with configurationSlot is <Configured configurationSlot> or <Configured configurationSlot2> depending on which one is already in use</p> <ul style="list-style-type: none"> - 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> <p><u>Note(s):</u></p> <ul style="list-style-type: none"> - The Test System checks the ActiveNetworkProfile variable to see which slot is currently active. - The Test System prevents overwriting the NetworkProfile at the active slot, as this is not recommended.
4. The Charging Station responds with a SetVariablesResponse	<p>3. The Test System sends a SetVariablesRequest with variable.name is "NetworkConfigurationPriority"</p> <p>component.name is "OCPPCommCtrlr"</p> <p>attributeValue is <configurationSlot set at Step 1, the other configured configurationSlot></p>
6. The Charging Station responds with a ResetResponse	<p>5. The Test System sends a ResetRequest with type OnIdle</p>

Main (Test scenario)	
7. The Charging Station connects to the configured alternative internal Test System endpoint. <u>Note(s):</u> - <i>During the TLS handshake the Charging Station validates and accepts the CSMS certificate, signed by the <Configured (new) CSMS Root certificate 2>.</i>	8. The connection attempt is accepted by the Test System.
9. Execute Reusable State <i>Booted</i>	

Tool validations
* Step 6: Message ResetResponse - status <i>Accepted</i>
Post scenario validations: - N/a

TC_B_51_CS: Status change during offline period - > Offline Threshold

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 Test System closes the WebSocket connection AND does not accept a reconnect.
2. <u>Manual Action</u> : Connect the EV and EVSE.	
	3. The Test System 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 Test System responds accordingly.

Tool validations
<p>* Step 4:</p> <p><i>Configured EVSE/Connector:</i></p> <p>Message: StatusNotificationRequest</p> <ul style="list-style-type: none">- evseld <Configured evseld>- connectorId <Configured connectorId>- for the connector involved in the transaction: connectorStatus <i>Occupied</i>- for other connectors of the same EVSE: connectorStatus <i>Available</i> or <i>Unavailable</i> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none">- eventData[0].trigger <i>Delta</i>- eventData[0].component.name <i>"Connector"</i>- eventData[0].variable.name <i>"AvailabilityState"</i>- eventData[0].evse.id <Configured evseld>- eventData[0].connectorId <Configured connectorId>- for the connector involved in the transaction: eventData[0].actualValue <i>Occupied</i>- for other connectors of the same EVSE: eventData[0].actualValue <i>Available</i> or <i>Unavailable</i> <p><i>All other EVSE/Connector(s):</i></p> <p>Message: StatusNotificationRequest</p> <ul style="list-style-type: none">- connectorStatus <i>Available</i> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none">- eventData[0].trigger <i>Delta</i>- eventData[0].actualValue <i>"Available"</i>- eventData[0].component.name <i>"Connector"</i>- eventData[0].variable.name <i>"AvailabilityState"</i>
<p>Post scenario validations:</p> <p>N/a</p>

TC_B_52_CS: Status change during offline period - < Offline Threshold

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:

OCPPEndpoint.OfflineThreshold is <Configured offlineThreshold>

OCPPEndpoint.RetryBackOffWaitMinimum is <Configured offlineThreshold> / 2

OCPPEndpoint.RetryBackOffRandomRange is 0

Memory State:

N/a

Reusable State(s):

N/a

Main (Test scenario)

Charging Station	CSMS
1. The Test System closes the WebSocket connection AND does not accept a reconnect.	
2. <u>Manual Action</u> : Connect the EV and EVSE.	
3. The Test System accepts reconnection attempt from the Charging Station.	
4. The Charging Station notifies the CSMS about the current state of the configured connector.	5. The Test System responds accordingly.

Tool validations

* Step 3:

Message: **StatusNotificationRequest**

- **evseld** <Configured evseld>

- **connectorId** <Configured connectorId>

- for the connector involved in the transaction: **connectorStatus** *Occupied*

- optionally for other connectors of the same EVSE: **connectorStatus** *Unavailable*

Message: **NotifyEventRequest**

- **eventData[0].trigger** *Delta*

- **eventData[0].component.name** "Connector"

- **eventData[0].variable.name** "AvailabilityState"

- **eventData[0].evse.id** <Configured evseld>

- **eventData[0].connectorId** <Configured connectorId>

- for the connector involved in the transaction: **eventData[0].actualValue** *Occupied*

- optionally for other connectors of the same EVSE: **eventData[0].actualValue** *Unavailable*

Post scenario validations:

N/a

TC_B_53_CS: Get Base Report - Test mandatory DM variables via FullInventory

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
<p>2. CS responds with:</p> <p>GetBaseReportResponse with status = <i>Accepted</i></p> <p>3. CS sends one or more NotifyReportRequest messages to report all its component/variables.</p>	<p>1. Test System requests a GetBaseReportRequest with:</p> <p>reportBase = <i>FullInventory</i> and</p> <p>requestId = <Generated requestId></p> <p>4. Test System responds with a NotifyReportResponse for each NotifyReportRequest</p>

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 Test System checks that all implemented standardized components / variables are implemented correctly according to the OCPP specification:

- 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**.
- For variables with **mutability** NOT set to *WriteOnly* the variableAttribute.value is NOT omitted in the **NotifyReportRequest**. There is one exception to this rule and that is for **EVSE.Power**. This variable only has a **maxLimit**.

TC_B_54_CS: Get Custom Report - with component/variable, but no instance

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. Test System sends GetReportRequest with: - requestId = <Generated requestId> - componentVariable.component.name = "DeviceDataCtrlr" - componentVariable.variable.name = "ItemsPerMessage"
3. Charging Station sends NotifyReportRequest	4. Test System responds with NotifyReportResponse

Tool validations
* Step 2: Message: GetReportResponse with: - status = <i>Accepted</i> - 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" <i>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.</i>
Post scenario validations: N/A

TC_B_55_CS: Get Custom Report - with component/variable/instance

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. Test System sends GetReportRequest with: - requestId = <Generated requestId> - componentVariable.component.name = "DeviceDataCtrlr" - componentVariable.variable.name = "ItemsPerMessage" - componentVariable.instance = "GetReport"
3. Charging Station sends NotifyReportRequest	4. Test System 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

TC_B_56_CS: Get Custom Report - with component/variable, but no evseld

Test case name	Get Custom Report - with component/variable, but no evseld
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 evseld 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. Test System sends GetReportRequest with: - requestId = <Generated requestId> - componentVariable.component.name = "EVSE" - componentVariable.variable.name = "AvailabilityState"
3. Charging Station sends NotifyReportRequest	4. Test System responds with NotifyReportResponse

Tool validations
* Step 2: Message: GetReportResponse with: - status = <i>Accepted</i> - 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> <i>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.</i>
Post scenario validations: N/A

TC_B_57_CS: Network Reconnection - After connection loss

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.NetworkProfileConnectionAttempts is 3

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 Test System closes the websocket connection.	
2. The Charging Station waits for the duration of the configured RetryBackOffWaitMinimum and reconnects to the Test System.	3. The connection attempt is accepted by the Test System.
4. The Test System closes the websocket connection.	
5. The Charging Station waits for the duration of the configured RetryBackOffWaitMinimum and reconnects to the Test System.	6. The connection attempt is not accepted by the Test System.
7. The Charging Station waits for double the reconnection time used at step 2 and reconnects to the Test System.	8. The connection attempt is accepted by the Test System.

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 2.

Post scenario validations:

- N/a

TC_B_100_CS: Set new NetworkConnectionProfile - Identity and password

Test case name	Set new NetworkConnectionProfile - Identity and password
Test case Id	TC_B_100_CS
Use case Id(s)	B09
Requirement(s)	B09.FR.01, B09.FR.08, B09.FR.09, B09.FR.11, B09.FR.12
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 properly reporting the network connection profile in the device model.
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 Test System sends a SetNetworkProfileRequest with: - configurationSlot is <Configured non-active configurationSlot> - connectionData.ocppVersion OCPP21 - connectionData.ocppInterface Any - connectionData.ocppTransport JSON - connectionData.messageTimeout <Configured messageTimeout> - connectionData.ocppCsmsUrl <Configured ocppCsmsUrl> - connectionData.securityProfile 2 - connectionData.identity <Configured identity> - connectionData.basicAuthPassword <Configured basicAuthPassword>
4. The Charging Station responds with GetBaseReportResponse	3. The Test System sends GetBaseReportRequest with requestId = <Generated requestId> reportBase = FullInventory
5. The Charging Station sends NotifyReportRequest	6. The Test System responds with NotifyReportResponse
<u>Note(s):</u> - If NotifyReportRequest.tbc is True in Step 5 then step 5 and 6 will be repeated	

Tool validations
* Step 2: Message SetNetworkProfileResponse - status must be Accepted * Step 4: Message GetBaseReportResponse - status must be Accepted

Tool validations

* Step 5:

Note 1: The value of **<reportData.variableCharacteristics.valuesList** of step 5> for **reportData.component.name** = OCPPCommCtrlr and **reportData.variable.name** = NetworkConfigurationPriority is referred to as **<supported configurationSlots>**

Note 2: Only the values for **reportData[*].component.name** = NetworkConfiguration are of interest to this test case. The order in which components and variables appear in the report is undefined.

Message **NotifyReportRequest**

- **requestId** = <Generated requestId>

FOR EACH <configurationSlot> IN <supported configurationSlots>

- **reportData[*].component.name** must be NetworkConfiguration
- **reportData[*].component.instance** must be <configurationSlot>
- **reportData[*].variable.name** must be OcppCsmsUrl
- IF <configurationSlot> = <Configured non-active configurationSlot> **reportData[*].variableAttribute.value** must be <Configured ocppCsmsUrl>

- **reportData[*].component.name** must be NetworkConfiguration
- **reportData[*].component.instance** must be <configurationSlot>
- **reportData[*].variable.name** must be OcppInterface
- IF <configurationSlot> = <Configured non-active configurationSlot> **reportData[*].variableAttribute.value** must be Any

- **reportData[*].component.name** must be NetworkConfiguration
- **reportData[*].component.instance** must be <configurationSlot>
- **reportData[*].variable.name** must be OcppTransport
- IF <configurationSlot> = <Configured non-active configurationSlot> **reportData[*].variableAttribute.value** must be JSON

- **reportData[*].component.name** must be NetworkConfiguration
- **reportData[*].component.instance** must be <configurationSlot>
- **reportData[*].variable.name** must be OcppVersion
- IF <configurationSlot> = <Configured non-active configurationSlot> **reportData[*].variableAttribute.value** must be OCPP21

- **reportData[*].component.name** must be NetworkConfiguration
- **reportData[*].component.instance** must be <configurationSlot>
- **reportData[*].variable.name** must be MessageTimeout
- IF <configurationSlot> = <Configured non-active configurationSlot> **reportData[*].variableAttribute.value** must be <Configured messageTimeout>

- **reportData[*].component.name** must be NetworkConfiguration
- **reportData[*].component.instance** must be <configurationSlot>
- **reportData[*].variable.name** must be SecurityProfile
- IF <configurationSlot> = <Configured non-active configurationSlot> **reportData[*].variableAttribute.value** must be 2

- **reportData[*].component.name** must be NetworkConfiguration
- **reportData[*].component.instance** must be <configurationSlot>
- **reportData[*].variable.name** must be BasicAuthPassword
- **reportData[*].variableAttribute.mutability** must be WriteOnly
- **reportData[*].variableAttribute.value** must be <omitted>

- **reportData[*].component.name** must be NetworkConfiguration
- **reportData[*].component.instance** must be <configurationSlot>
- **reportData[*].variable.name** must be VpnEnabled
- IF <configurationSlot> = <Configured non-active configurationSlot> **reportData[*].variableAttribute.value** must be false

Tool validations
<ul style="list-style-type: none">- reportData[*].component.name must be <i>NetworkConfiguration</i>- reportData[*].component.instance must be <i><configurationSlot></i>- reportData[*].variable.name must be <i>ApnEnabled</i>- IF <i><configurationSlot> = <Configured non-active configurationSlot></i> reportData[*].variableAttribute.value must be <i>false</i>
END FOR EACH
Post scenario validations: <ul style="list-style-type: none">- N/a

TC_B_101_CS: Reset ImmediateAndResume - TxResumptionTimeout 0 or <omitted> - Rejected

Test case name	Reset ImmediateAndResume - With Ongoing Transaction - TxResumptionTimeout 0 or <omitted> - Rejected
Test case Id	TC_B_101_CS
Use case Id(s)	B13
Requirement(s)	B13.FR.01
System under test	Charging Station
Description	The CSMS can remotely request the Charging Station to reset itself by sending a ResetRequest during a transaction. When ResetRequest "ImmediateAndResume" is sent the charging stations reboots and resumes transactions after reboot.
Purpose	To verify if the Charging Station is able to reject a ResetRequest with type ImmediateAndResume when it does not support the mechanism to resume a transaction after a reset.
Prerequisite(s)	n/a

Before (Preparations)
Configuration State: TxCtrlr.ResumptionTimeout = 0 (If implemented)
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 Test System sends a ResetRequest with type ImmediateAndResume

Tool validations
* Step 2: Message ResetResponse - status = <i>Rejected</i>
Post scenario validations: - N/a

TC_B_102_CS: Reset ImmediateAndResume - With ongoing transaction - Energy Transfer Suspended

Test case name	Reset ImmediateAndResume - With ongoing transaction - Energy Transfer Suspended
Test case Id	TC_B_102_CS
Use case Id(s)	B13
Requirement(s)	B13.FR.03, B13.FR.04, B13.FR.20, B13.FR.23, B13.FR.25, B13.FR.30
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 "ImmediateAndResume" is sent the charging stations reboots and resumes transactions after reboot.
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)	The Charging Station supports transaction resumption after a reboot.

Before (Preparations)
Configuration State: TxCtrlr.ResumptionTimeout = 999 TxCtrlr.AllowEnergyTransferResumption = false
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>Booted</i> with reset type <i>ImmediateAndResume</i>	
2. The Charging Station sends a TransactionEventRequest <i>This step is optional: In some cases, a Charging Station might re-detect the cable being plugged in after a reboot.</i>	3. The Test System responds with a TransactionEventResponse
4. The Charging Station sends a TransactionEventRequest	5. The Test System responds with a TransactionEventResponse

Tool validations
<p>* Step 2:</p> <p>Message TransactionEventRequest</p> <ul style="list-style-type: none"> - eventType = <i>Updated</i> - triggerReason = <i>CablePluggedIn</i> - transactionInfo.transactionId <transactionId remained the same> <p>* Step 4:</p> <p>Message TransactionEventRequest</p> <ul style="list-style-type: none"> - eventType = <i>Updated</i> - triggerReason = <i>TxResumed</i> - transactionInfo.chargingState <i>SuspendedEVSE</i> - transactionInfo.transactionId <transactionId remained the same>
Post scenario validations: - N/a

TC_B_103_CS: Reset ImmediateAndResume - With Ongoing Transaction - Resuming Energy Transfer

Test case name	Reset ImmediateAndResume - With Ongoing Transaction - Resuming Energy Transfer
Test case Id	TC_B_103_CS
Use case Id(s)	B13
Requirement(s)	B13.FR.03, B13.FR.04, B13.FR.20, B13.FR.23, B13.FR.24, B13.FR.30
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 "ImmediateAndResume" is sent the charging stations reboots and resumes transactions after reboot.
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)	The Charging Station supports transaction and energy transfer resumption after a reboot.

Before (Preparations)
Configuration State: TxCtrlr.ResumptionTimeout = 999 TxCtrlr.AllowEnergyTransferResumption = true
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>Booted</i> with reset type <i>ImmediateAndResume</i>	
2. The Charging Station sends a TransactionEventRequest <i>This step is optional: In some cases, a Charging Station might re-detect the cable being plugged in after a reboot.</i>	3. The Test System responds with a TransactionEventResponse
4. The Charging Station sends a TransactionEventRequest	5. The Test System responds with a TransactionEventResponse

Tool validations
* Step 2: Message TransactionEventRequest - eventType = <i>Updated</i> - triggerReason = <i>CablePluggedIn</i> - transactionInfo.transactionId <transactionId remained the same> * Step 4: Message TransactionEventRequest - eventType = <i>Updated</i> - triggerReason = <i>TxResumed</i> - transactionInfo.chargingState <i>Charging</i> - transactionInfo.transactionId <transactionId remained the same>
Post scenario validations: - N/a

TC_B_104_CS: Reset ImmediateAndResume - Without ongoing transaction

Test case name	Reset ImmediateAndResume - Without ongoing transaction
Test case Id	TC_B_104_CS
Use case Id(s)	B13
Requirement(s)	B13.FR.02
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 with no running transactions. When ResetRequest "ImmediateAndResume" is sent the charging station reboots when ResumptionTimeOut > 0.
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)	Charging Station has support for resuming transactions

Before (Preparations)
Configuration State: TxCtrlr.ResumptionTimeout = <Configured ResumptionTimeout>
Memory State: N/a
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>Booted</i> with reset type <i>ImmediateAndResume</i>	

Tool validations
N/a
Post scenario validations: N/a

TC_B_105_CS: Set new NetworkConnectionProfile - Add new NetworkConfiguration using SetVariables

Test case name	Set new NetworkConnectionProfile - Add new NetworkConfiguration using SetVariables
Test case Id	TC_B_105_CS
Use case Id(s)	B09
Requirement(s)	B09.FR.10, B09.FR.23
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 set the network connection profile via component NetworkConfiguration.
Prerequisite(s)	NetworkConfiguration component variables are writable.

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

Main (Test scenario)	
Charging Station	CSMS
<u>Note:</u> In the following steps <configurationSlot> equals <Configured non-active configurationSlot>	
2. The Charging Station responds with GetVariablesResponse	1. Test System sends a GetVariablesRequest for - DeviceDataCtrlr.ItemsPerMessage[SetVariables]
<u>Note:</u> In the following steps the notation for setting a variable, like: NetworkConfiguration[<instance>].OcppCsmsUrl = <value> expands to: - setVariableData[0].component.name <i>NetworkConfiguration</i> - setVariableData[0].component.instance <i><instance></i> - setVariableData[0].variable.name <i>OcppCsmsUrl</i> - setVariableData[0].attributeValue <i><value></i>	
<u>Note:</u> If the value of DeviceDataCtrlr.ItemsPerMessage[SetVariables] from step #1 is 9 or more , than the following can all be set in one SetVariablesRequest, otherwise the variables are set one-by-one.	
4. The Charging Station responds with SetVariablesResponse(s)	3. Test System sends a SetVariablesRequest on component NetworkConfiguration[<Configured non-active configurationSlot>] for: - OcppCsmsUrl = <i><Configured ocppCsmsUrl></i> - OcppInterface = <i>Any</i> - OcppTransport = <i>JSON</i> - OcppVersion = <i>OCPP21</i> - MessageTimeout = <i><Configured messageTimeout></i> - SecurityProfile = <i><Active securityProfile></i> - Identity = <i><Configured charging station identity></i> - BasicAuthPassword = <i>PasswordOfSufficientLength</i> - VpnEnabled = <i>false</i> - ApnEnabled = <i>false</i>

Tool validations
<p>* Step 2:</p> <p>Message GetVariablesResponse</p> <ul style="list-style-type: none">- getVariableResult.attributeStatus must be <i>Accepted</i>
<p>* Step 4:</p> <p>Message SetVariablesResponse</p> <p><i>for all SetVariablesResponses except VpnEnabled, ApnEnabled:</i></p> <ul style="list-style-type: none">- setVariableResult.attributeStatus must be <i>Accepted</i> <p><i>for SetVariablesResponses for VpnEnabled/ApnEnabled:</i></p> <ul style="list-style-type: none">- setVariableResult.attributeStatus must be <i>Accepted</i> or <i>Rejected</i>
<p>Post scenario validations:</p> <ul style="list-style-type: none">- N/a

TC_B_107_CS: Set new NetworkConnectionProfile - Add and remove slot from NetworkConfigurationPriority

Test case name	Set new NetworkConnectionProfile - Add and remove slot from NetworkConfigurationPriority
Test case Id	TC_B_107_CS
Use case Id(s)	B09
Requirement(s)	B09.FR.20, B09.FR.33, B09.FR.34
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 add and remove a slot from NetworkConfigurationPriority and check the associated configuration values in NetworkConfiguration when adding.
Prerequisite(s)	NetworkConfiguration component variables are writable.

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 GetVariablesResponse	1. Test System sends GetVariablesRequest for OCPPCommCtrlr.NetworkConfigurationPriority.
<u>Note:</u> - If value in step #2 contains the non-active networkProfile slot, then step 3 and 4 are executed. - If value in step #2 does NOT contain the non-active networkProfile slot, then similar steps will be executed at the end of the testcase; step 17 and 18.	
4. Charging Station responds with SetVariablesResponse	3. Test System sends SetVariablesRequest for OCPPCommCtrlr.NetworkConfigurationPriority with: - setVariableData.attributeValue has value of NetworkConfigurationPriority with <Configured non-active configurationSlot> removed.
<u>Note:</u> If value in step #2 was "1,2" and non-active slot is 2, then the new value in step #3 will be "1"	
6. The Charging Station responds with GetVariablesResponse	5. Test System sends a GetVariablesRequest for - DeviceDataCtrlr.ItemsPerMessage[GetVariables]
<u>Note:</u> In the following steps the notation for setting a variable, like: NetworkConfiguration[<instance>].OcppCsmsUrl = <value> expands to: - setVariableData[0].component.name NetworkConfiguration - setVariableData[0].component.instance <instance> - setVariableData[0].variable.name OcppCsmsUrl - setVariableData[0].attributeValue <value>	
<u>Note:</u> If the value of DeviceDataCtrlr.ItemsPerMessage[GetVariables] from step #1 is 9 or more , than the following can all be set in one SetVariablesRequest, otherwise the variables are set one-by-one.	

Main (Test scenario)	
8. The Charging Station responds with SetVariablesResponse(s)	<p>7. Test System sends a SetVariablesRequest on component NetworkConfiguration[<Configured non-active configurationSlot>] for:</p> <ul style="list-style-type: none"> - OcppCsmsUrl = <i>ws://invalidurl.example.com/</i> - OcppInterface = <i>Any</i> - OcppTransport = <i>JSON</i> - OcppVersion = <i>OCPP21</i> - MessageTimeout = <i><Configured messageTimeout></i> - SecurityProfile = <i>2</i> - BasicAuthPassword = <i>PasswordOfSufficientLength</i> - VpnEnabled = <i>false</i> - ApnEnabled = <i>false</i>
10. Charging Station responds with SetVariablesResponse	<p>9. Test System sends SetVariablesRequest for OCPPCommCtrlr.NetworkConfigurationPriority with:</p> <ul style="list-style-type: none"> - setVariableData.attributeValue with <i><Configured non-active configurationSlot></i> added at the end.
12. The Charging Station responds with SetVariablesResponses	<p>11. Test System sends a SetVariablesRequest on component NetworkConfiguration[<Configured non-active configurationSlot>] for:</p> <ul style="list-style-type: none"> - OcppCsmsUrl = <i>wss://validurl.example.com:443/</i>
14. Charging Station responds with SetVariablesResponse	<p>13. Test System sends SetVariablesRequest for OCPPCommCtrlr.NetworkConfigurationPriority with:</p> <ul style="list-style-type: none"> - setVariableData.attributeValue with <i><Configured non-active configurationSlot></i> added at the end.
16. Charging Station responds with GetVariablesResponse	<p>15. Test System sends GetVariablesRequest for OCPPCommCtrlr.NetworkConfigurationPriority.</p>
<p>Note:</p> <ul style="list-style-type: none"> - Remove <i><Configured non-active configurationSlot></i> from <i>NetworkConfigurationPriority</i>. 	
18. Charging Station responds with SetVariablesResponse	<p>17. Test System sends SetVariablesRequest for OCPPCommCtrlr.NetworkConfigurationPriority with:</p> <ul style="list-style-type: none"> - setVariableData.attributeValue has one slot less than reported in step #16. (Last slot removed).
<p>Note: If value in step #16 was "1,2" then the new value in step #17 will be "1"</p>	

Tool validations
<p>* Step 4:</p> <p>Message SetVariablesResponse</p> <ul style="list-style-type: none"> - setVariableResult.attributeStatus must be <i>Accepted</i>
<p>* Step 8:</p> <p>Message SetVariablesResponse</p> <p>for all <i>SetVariablesResponses</i> except <i>VpnEnabled, ApnEnabled</i></p> <ul style="list-style-type: none"> - setVariableResult.attributeStatus must be <i>Accepted</i> <p>for <i>SetVariablesResponses</i> for <i>VpnEnabled, ApnEnabled</i>:</p> <ul style="list-style-type: none"> - setVariableResult.attributeStatus must be <i>Accepted</i> or <i>Rejected</i>
<p>* Step 10:</p> <p>Message SetVariablesResponse</p> <ul style="list-style-type: none"> - setVariableResult.attributeStatus must be <i>Rejected</i> - setVariableResult.attributeStatus.reasonCode must be <i>"InvalidNetworkConf"</i>
<p>* Step 12:</p> <p>Message SetVariablesResponse</p> <ul style="list-style-type: none"> - setVariableResult.attributeStatus must be <i>Accepted</i>
<p>* Step 14:</p> <p>Message SetVariablesResponse</p> <ul style="list-style-type: none"> - setVariableResult.attributeStatus must be <i>Accepted</i>

Tool validations
* Step 18: Message SetVariablesResponse - setVariableResult.attributeStatus must be <i>Accepted</i>
Post scenario validations: - N/a

TC_B_108_CS: Set new NetworkConnectionProfile - Prevent overwriting configured Network Profile slot

Test case name	Set new NetworkConnectionProfile - Prevent overwriting configured Network Profile slot
Test case Id	TC_B_108_CS
Use case Id(s)	B09
Requirement(s)	B09.FR.22
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 that the Charging Station does not allow to change the configuration of a network slot that is in NetworkConfigurationPriority, which may be used during reconnection.
Prerequisite(s)	NetworkConfiguration component variables are writable.

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. Test System sends GetVariablesRequest for - OCPPCommCtrlr.NetworkConfigurationPriority
<i>Note: For each value in NetworkConfigurationPriority, try to set a NetworkConfiguration instance:</i>	
FOR slot in attributeValue of OCPPCommCtrlr.NetworkConfigurationPriority DO	
4. Charging Station responds with SetVariablesResponse	3. Test System sends SetVariablesRequest with: - setVariableData[0].component.name NetworkConfiguration - setVariableData[0].component.instance <slot> - setVariableData[0].variable.name MessageTimeout - setVariableData[0].attributeValue 123
END FOR	

Tool validations
<p>* Step 2:</p> <p>Message GetVariablesResponse with:</p> <ul style="list-style-type: none"> - getVariableResult[0].component = OCPPCommCtrlr - getVariableResult[0].variable = NetworkConfigurationPriority - getVariableResult[0].attributeValue = <a list of at least two values> - getVariableResult[0].attributeStatus = Accepted

Tool validations
<p>* Step 4:</p> <p>FOR <i>slot</i> in OCPPCommCtrlr.NetworkConfigurationPriority DO</p> <p>Message SetVariablesResponse with:</p> <ul style="list-style-type: none">- setVariableResult[0].component.name <i>NetworkConfiguration</i>- setVariableResult[0].component.instance <i><slot></i>- setVariableResult[0].variable.name <i>MessageTimeout</i>- setVariableResult[0].attributeStatus <i>Rejected</i>- setVariableResult[0].attributeStatusInfo.reasonCode = <i>PriorityNetworkConf</i> <p>END FOR</p>
<p>Post scenario validations:</p> <ul style="list-style-type: none">- N/a

TC_B_109_CS: Set new NetworkConnectionProfile - When changing SecurityCtrlr.Identity/BasicAuthPassword the NetworkProfiles.Identity/BasicAuthPassword must be cleared

Test case name	Set new NetworkConnectionProfile - When changing SecurityCtrlr.Identity/BasicAuthPassword the NetworkProfiles.Identity/BasicAuthPassword must be cleared
Test case Id	TC_B_109_CS
Use case Id(s)	B09
Requirement(s)	B09.FR.16, B09.FR.18, B09.FR.26, B09.FR.27
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 that the Charging Station clears Identity/BasicAuthPassword from the active network connection profile, when this value is set via SecurityCtrlr component. Clearing of BasicAuthPassword cannot be checked, because this field is not readable.
Prerequisite(s)	NetworkConfiguration component variables are writable. NetworkConfiguration.Identity and NetworkConfiguration.BasicAuthPassword are set.

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. Test System sends GetVariablesRequest for - OCCPPCommCtrlr.ActiveNetworkProfile
<u>Note:</u> The result of Step #2 is called <the active configuration>	
4. The Charging Station responds with GetVariablesResponse	3. Test System sends GetVariablesRequest for - NetworkConfiguration[<the active configuration>].Identity
6. The Charging Station responds with SetVariablesResponses	5. Test System sends SetVariablesRequests for - SecurityCtrlr.Identity = <configured ChargingStationId> - SecurityCtrlr.BasicAuthPassword = <configured BasicAuthPassword>
8. The Charging Station responds with GetVariablesResponse	7. Test System sends GetVariablesRequest for - NetworkConfiguration[<the active configuration>].Identity
10. Charging Station reconnects	9. Test System closes the websocket connection
<u>Note:</u> Test System reconnects using values from SecurityCtrlr.Identity/BasicAuthPassword	
12. The Charging Station responds with GetVariablesResponse	11. Test System sends GetVariablesRequest for - OCCPPCommCtrlr.ActiveNetworkProfile
<u>Note:</u> Restoring the Identity and BasicAuthPassword via SetNetworkProfileRequest and reconnecting.	
14. The Charging Station responds with GetVariablesResponse	13. Test System sends GetVariablesRequest for - NetworkConfiguration[<the active configuration>].SecurityProfile

Main (Test scenario)	
16 The Charging Station responds with SetNetworkProfileResponse	15 The Test System sends a SetNetworkProfileRequest with: <ul style="list-style-type: none"> - configurationSlot is <i><the active configuration></i> - connectionData.ocppVersion <i>OCPP21</i> - connectionData.ocppInterface <i>Any</i> - connectionData.ocppTransport <i>JSON</i> - connectionData.messageTimeout <i><Configured messageTimeout></i> - connectionData.ocppCsmsUrl <i><Configured ocppCsmsUrl></i> - connectionData.securityProfile <i><security profile from step 14></i> - connectionData.identity = <i><Configured ChargingStationId></i> - connectionData.basicAuthPassword <i><Configured basicAuthPassword></i>
18. Charging Station reconnects	17. Test System closes the websocket connection
20. The Charging Station responds with GetVariablesResponse	19. Test System sends GetVariablesRequest for <ul style="list-style-type: none"> - OCPPCommCtrlr.ActiveNetworkProfile

Tool validations
<p>* Step 2:</p> <p>Messages GetVariableResponse with:</p> <ul style="list-style-type: none"> - getVariableResult[0].component = <i>OCPPCommCtrlr</i> - getVariableResult[0].variable = <i>ActiveNetworkProfile</i> - getVariableResult[0].attributeValue = <i><any configuration slot></i> - getVariableResult[0].attributeStatus = <i>Accepted</i>
<p>* Step 4:</p> <p>Messages GetVariableResponse with:</p> <ul style="list-style-type: none"> - getVariableResult[0].component = <i>NetworkConfiguration</i> - getVariableResult[0].instance = <i><the active configuration></i> - getVariableResult[0].variable = <i>Identity</i> - getVariableResult[0].attributeValue = <i><identity of charging station></i> - getVariableResult[0].attributeStatus = <i>Accepted</i>
<p>* Step 6:</p> <p>Messages SetVariableResponse with:</p> <ul style="list-style-type: none"> - setVariableResult[0].component.name = <i>SecurityCtrlr</i> - setVariableResult[0].variable.name = <i>Identity</i> - setVariableResult[0].attributeStatus = <i>Accepted</i> <ul style="list-style-type: none"> - setVariableResult[1].component.name = <i>SecurityCtrlr</i> - setVariableResult[1].variable.name = <i>BasicAuthPassword</i> - setVariableResult[1].attributeStatus = <i>Accepted</i>
<p>* Step 8:</p> <p>Messages GetVariableResponse with:</p> <ul style="list-style-type: none"> - getVariableResult[0].component = <i>NetworkConfiguration</i> - getVariableResult[0].variable = <i>Identity</i> - getVariableResult[0].attributeValue = <i><empty></i> - getVariableResult[0].attributeStatus = <i>Accepted</i>
<p>* Step 12:</p> <p>Messages GetVariableResponse with:</p> <ul style="list-style-type: none"> - getVariableResult[0].component = <i>OCPPCommCtrlr</i> - getVariableResult[0].variable = <i>ActiveNetworkProfile</i> - getVariableResult[0].attributeValue = <i><the active configuration></i> - getVariableResult[0].attributeStatus = <i>Accepted</i>

Tool validations
<p>* Step 14:</p> <p>Messages GetVariableResponse with:</p> <ul style="list-style-type: none">- getVariableResult[0].component = <i>NetworkConfiguration</i>- getVariableResult[0].variable = <i>SecurityProfile</i>- getVariableResult[0].attributeValue not <i><omitted></i>- getVariableResult[0].attributeStatus = <i>Accepted</i>
<p>* Step 16:</p> <p>Messages SetNetworkConfigurationResponse with:</p> <ul style="list-style-type: none">- status = <i>Accepted</i>
<p>* Step 20:</p> <p>Messages GetVariableResponse with:</p> <ul style="list-style-type: none">- getVariableResult[0].component = <i>OCPPCommCtrlr</i>- getVariableResult[0].variable = <i>ActiveNetworkProfile</i>- getVariableResult[0].attributeValue = <i><the active configuration></i>- getVariableResult[0].attributeStatus = <i>Accepted</i>
<p>Post scenario validations:</p> <ul style="list-style-type: none">- N/a

TC_B_110_CS: Set new NetworkConnectionProfile - No security downgrade to profile #1

Test case name	Set new NetworkConnectionProfile - No security downgrade to profile #1
Test case Id	TC_B_110_CS
Use case Id(s)	B09
Requirement(s)	B09.FR.31
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 that the Charging Station refuses to downgrade to security profile #1 via SetNetworkProfileRequest.
Prerequisite(s)	Current security profile is higher than 1.

Before (Preparations)
Configuration State: SecurityCtrlr.AllowSecurityProfileDowngrade = true
Memory State: N/a
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetNetworkProfileResponse	<p>1. The Test System sends a SetNetworkProfileRequest with:</p> <ul style="list-style-type: none">- configurationSlot is <Configured non-active configurationSlot>- connectionData.ocppVersion OCPP21- connectionData.ocppInterface Any- connectionData.ocppTransport JSON- connectionData.messageTimeout <Configured messageTimeout>- connectionData.ocppCsmsUrl <Configured ocppCsmsUrl>- connectionData.securityProfile 1- connectionData.basicAuthPassword <Configured basicAuthPassword>

Tool validations
<p>* Step 2:</p> <p>Message SetNetworkProfileResponse with:</p> <ul style="list-style-type: none"> - status = Rejected - statusInfo.reasonCode = "NoSecurityDowngrade"
Post scenario validations:
- N/a

TC_B_111_CS: Set new NetworkConnectionProfile - No security downgrade to profile #1 - DM

Test case name	Set new NetworkConnectionProfile - No security downgrade to profile #1 - DM
Test case Id	TC_B_111_CS
Use case Id(s)	B09
Requirement(s)	B09.FR.32
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 that the Charging Station refuses to downgrade security profile #1 via the NetworkConfiguration component.
Prerequisite(s)	Current security profile is higher than 1. The variables of component NetworkConfiguration are writeable.

Before (Preparations)
Configuration State: SecurityCtrlr.AllowSecurityProfileDowngrade = <i>true</i>
Memory State: N/a
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
<u>Note:</u> In the following steps <configurationSlot> equals <Configured non-active configurationSlot>	
2. The Charging Station responds with GetVariablesResponse	1. Test System sends a GetVariablesRequest for - DeviceDataCtrlr.ItemsPerMessage[GetVariables]
<u>Note:</u> In the following steps the notation for setting a variable, like: NetworkConfiguration[<instance>].OcppCsmsUrl = <value> expands to: - setVariableData[0].component.name <i>NetworkConfiguration</i> - setVariableData[0].component.instance <instance> - setVariableData[0].variable.name <i>OcppCsmsUrl</i> - setVariableData[0].attributeValue <value>	
<u>Note:</u> If the value of DeviceDataCtrlr.ItemsPerMessage[GetVariables] from step #1 is 9 or more , than the following can all be set in one SetVariablesRequest, otherwise the variables are set one-by-one.	
4. The Charging Station responds with SetVariablesResponse(s)	3. Test System sends a SetVariablesRequest on component NetworkConfiguration[<Configured non-active configurationSlot>] for: - OcppCsmsUrl = <Configured ocppCsmsUrl> - OcppInterface = Any - OcppTransport = JSON - OcppVersion = OCPP21 - MessageTimeout = <Configured messageTimeout> - SecurityProfile = 1 - BasicAuthPassword = PasswordOfSufficientLength - VpnEnabled = false - ApnEnabled = false

Tool validations
<p>* Step 4:</p> <p>Message SetVariablesResponse with setVariableResult.variable = "SecurityProfile" has:</p> <ul style="list-style-type: none">- setVariableResult.attributeStatus = <i>Rejected</i>- setVariableResult.attributeStatusInfo.reasonCode = "NoSecurityDowngrade"
<p>Post scenario validations:</p> <ul style="list-style-type: none">- N/a

TC_B_112_CS: Set new NetworkConnectionProfile - AllowSecurityDowngrade is false

Test case name	Set new NetworkConnectionProfile - AllowSecurityDowngrade is false
Test case Id	TC_B_112_CS
Use case Id(s)	B09
Requirement(s)	B09.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 that the Charging Station refuses to downgrade security at all via SetNetworkProfileRequest.
Prerequisite(s)	Current security profile is higher than 2.

Before (Preparations)
Configuration State: SecurityCtrlr.AllowSecurityProfileDowngrade = <i>false</i>
Memory State: N/a
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetNetworkProfileResponse	<p>1. The Test System sends a SetNetworkProfileRequest with:</p> <ul style="list-style-type: none">- configurationSlot is <i><Configured non-active configurationSlot></i>- connectionData.ocppVersion <i>OCPP21</i>- connectionData.ocppInterface <i>Any</i>- connectionData.ocppTransport <i>JSON</i>- connectionData.messageTimeout <i><Configured messageTimeout></i>- connectionData.ocppCsmsUrl <i><Configured ocppCsmsUrl></i>- connectionData.securityProfile <i>2</i>- connectionData.basicAuthPassword <i><Configured basicAuthPassword></i>

Tool validations
<p>* Step 2:</p> <p>Message SetNetworkProfileResponse with:</p> <ul style="list-style-type: none"> - status = <i>Rejected</i> - statusInfo.reasonCode = "NoSecurityDowngrade"
Post scenario validations:
- N/a

TC_B_113_CS: Set new NetworkConnectionProfile - AllowSecurityDowngrade = false - DM

Test case name	Set new NetworkConnectionProfile - AllowSecurityDowngrade = false - DM
Test case Id	TC_B_113_CS
Use case Id(s)	B09
Requirement(s)	B09.FR.35
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 that the Charging Station refuses to downgrade security at all via the NetworkConfiguration component.
Prerequisite(s)	Current security profile is higher than 2. The variables of component NetworkConfiguration are writeable.

Before (Preparations)
Configuration State: SecurityCtrlr.AllowSecurityProfileDowngrade = false
Memory State: N/a
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
<u>Note:</u> In the following steps <configurationSlot> equals <Configured non-active configurationSlot>	
2. The Charging Station responds with GetVariablesResponse	1. Test System sends a GetVariablesRequest for - DeviceDataCtrlr.ItemsPerMessage[GetVariables]
<u>Note:</u> In the following steps the notation for setting a variable, like: NetworkConfiguration[<instance>].OcppCsmsUrl = <value> expands to: - setVariableData[0].component.name <i>NetworkConfiguration</i> - setVariableData[0].component.instance <i><instance></i> - setVariableData[0].variable.name <i>OcppCsmsUrl</i> - setVariableData[0].attributeValue <i><value></i>	
<u>Note:</u> If the value of DeviceDataCtrlr.ItemsPerMessage[GetVariables] from step #1 is 9 or more , than the following can all be set in one SetVariablesRequest, otherwise the variables are set one-by-one.	
4. The Charging Station responds with SetVariablesResponse(s)	3. Test System sends a SetVariablesRequest on component NetworkConfiguration[<Configured non-active configurationSlot>] for: - OcppCsmsUrl = <i><Configured ocppCsmsUrl></i> - OcppInterface = <i>Any</i> - OcppTransport = <i>JSON</i> - OcppVersion = <i>OCPP21</i> - MessageTimeout = <i><Configured messageTimeout></i> - SecurityProfile = <i>2</i> - BasicAuthPassword = <i>PasswordOfSufficientLength</i> - VpnEnabled = <i>false</i> - ApnEnabled = <i>false</i>

Tool validations
<p>* Step 4:</p> <p>Message SetVariablesResponse with setVariableResult.variable = "SecurityProfile" has:</p> <ul style="list-style-type: none">- setVariableResult.attributeStatus = <i>Rejected</i>- setVariableResult.attributeStatusInfo.reasonCode = "NoSecurityDowngrade"
<p>Post scenario validations:</p> <ul style="list-style-type: none">- N/a

C Authorization

TC_C_02_CS: Local start transaction - Authorization Invalid/Unknown

Test case name	Local start transaction - Authorization Invalid/Unknown
Test case Id	TC_C_02_CS
Use case Id(s)	C01 OR C04 OR C06
Requirement(s)	C01.FR.02 OR C06.FR.02
System under test	Charging Station
Description	When a Charging Station needs to charge an EV, it needs to authorize the EV Driver first at the CSMS before the charging can be started or stopped.
Purpose	To verify whether the Charging Station is able to handle receiving an invalid idToken.
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station supports at least one authorization method described at the following Use cases; C01, C04, C06. - The Charging Station does NOT have a cable lock, which prevents the EV driver to connect the EV and EVSE before authorization.

Before (Preparations)

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
<u>Manual Action</u> : Present idToken.	
1. The Charging Station sends an AuthorizeRequest	2. The Test System responds with an AuthorizeResponse with idTokenInfo.status <i>Invalid</i>
<u>Note(s)</u> : <ul style="list-style-type: none"> - The Charging Station SHALL NOT send a TransactionEventRequest message after the AuthorizeRequest from step 1. - The Test System 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

TC_C_05_CS: Local start transaction - Authorization invalid - Cable lock

Test case name	Local start transaction - Authorization invalid - Cable lock
Test case Id	TC_C_05_CS
Use case Id(s)	C01 OR C04 OR C06
Requirement(s)	C01.FR.02 OR C06.FR.02
System under test	Charging Station
Description	When a Charging Station needs to charge an EV, it needs to authorize the EV Driver first before the charging can be started or stopped.
Purpose	To verify whether a Charging Station with a cable lock, which prevents the EV driver to connect the EV and EVSE before authorization, is able to handle receiving an invalid idToken.
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station has a cable lock, which prevents the EV driver to connect the EV and EVSE before authorization. - The Charging Station supports at least one authorization method described at the following Use cases; C01, C04, C06. - The Charging Station does NOT have the following configuration; TxStartPoint ReadOnly AND value Authorized is NOT set.

Before (Preparations)
Configuration State: AuthCtrlr.AuthEnabled is <i>true</i> (If implemented AND ReadWrite) AuthCtrlr.DisableRemoteAuthorization is <i>false</i> (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	2. The Test System responds with an AuthorizeResponse with idTokenInfo.status <i>Invalid</i>
<u>Note(s):</u> - The Charging Station SHALL NOT send a TransactionEventRequest message after the AuthorizeRequest from step 1. - The Test System 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

TC_C_04_CS: Local Stop Transaction - Different idToken

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)	<ul style="list-style-type: none"> - The Charging Station supports at least one authorization method described at the following Use cases; C01, C04. - The Charging Station does NOT use one idToken reader for multiple EVSE. - The Charging Station supports authorization methods other than NoAuthorization

Before (Preparations)
Configuration State: AuthCtrlr.AuthEnabled is <i>true</i> (If implemented AND ReadWrite) AuthCtrlr.DisableRemoteAuthorization is <i>false</i> (If implemented)
Memory State: <ul style="list-style-type: none"> - 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 <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action</u> : Present a different idToken than used to start the transaction.	
1. The Charging Station sends an AuthorizeRequest	2. The Test System responds with an AuthorizeResponse with idTokenInfo.status Accepted
<u>Note(s)</u> : - 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 Test System waits <Configured message timeout> seconds, before ending the testcase.	

Tool validations
N/a
Post scenario validations: <ul style="list-style-type: none"> - Charging Station has not sent a TransactionEventRequest(<i>Ended</i>).

TC_C_06_CS: Local start transaction - Authorization Blocked

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 <i>true</i> (If implemented AND ReadWrite) AuthCtrlr.DisableRemoteAuthorization is <i>false</i> (If implemented)
Memory State: N/a
Reusable State(s): State is <i>EVConnectedPreSession</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action:</u> Present idToken.	
1. The Charging Station sends an AuthorizeRequest	2. The Test System responds with an AuthorizeResponse with idTokenInfo.status <i>Blocked</i>
<u>Note(s):</u> - The Charging Station SHALL NOT send a TransactionEventRequest message after the AuthorizeRequest from step 7. - The Test System 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

TC_C_07_CS: Local start transaction - Authorization Expired

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 <i>true</i> (If implemented AND ReadWrite) AuthCtrlr.DisableRemoteAuthorization is <i>false</i> (If implemented)
Memory State: N/a
Reusable State(s): State is <i>EVConnectedPreSession</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action</u> : Present idToken.	
1. The Charging Station sends an AuthorizeRequest	2. The Test System responds with an AuthorizeResponse with idTokenInfo.status <i>Expired</i>
<u>Note(s)</u> : - The Charging Station SHALL NOT send a TransactionEventRequest message after the AuthorizeRequest from step 7. - The Test System 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

TC_C_08_CS: Authorization through authorization cache - Accepted

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)	The Charging Station has an authorization cache AND the Charging Station supports at least one of the following local start authorization options: C-30, C-31, C-32, C-34 OR supports at least one of the following remote start authorization options: C-36, C-37

Before (Preparations)
Configuration State: AuthCacheCtrlr.AuthCacheEnabled is <i>true</i> (If implemented) AuthCtrlr.LocalPreAuthorize is <i>true</i> (If implemented)
Memory State: <i>IdTokenCached</i> for <Configured valid IdToken fields>
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>Authorized</i> (Cached idToken)	
2. Execute Reusable State <i>EVConnectedPreSession</i>	
3. Execute Reusable State <i>EnergyTransferStarted</i>	

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.

TC_C_09_CS: Authorization through authorization cache - Invalid & Not Accepted

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)	The Charging Station has an authorization cache AND the Charging Station supports at least one of the following local start authorization options: C-30, C-31, C-32, C-34 OR supports at least one of the following remote start authorization options: C-36, C-37

Before (Preparations)
Configuration State: AuthCacheCtrlr.AuthCacheEnabled is <i>true</i> (If implemented) AuthCtrlr.LocalPreAuthorize is <i>true</i> (If implemented) AuthCacheCtrlr.DisablePostAuthorize is <i>false</i> (If implemented)
Memory State: <i>IdTokenCached</i> for <Configured invalid IdToken fields>
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>Authorized</i> for <Configured invalid IdToken fields>	

Tool validations
N/a
Post scenario validations: - N/a

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

TC_C_10_CS: Authorization through authorization cache - Blocked

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)	The Charging Station has an authorization cache AND the Charging Station supports at least one of the following local start authorization options: C-30, C-31, C-32, C-34 OR supports at least one of the following remote start authorization options: C-36, C-37

Before (Preparations)
Configuration State: AuthCacheCtrlr.AuthCacheEnabled is <i>true</i> (If implemented) AuthCtrlr.LocalPreAuthorize is <i>true</i> (If implemented) AuthCacheCtrlr.DisablePostAuthorize is <i>false</i> (If implemented)
Memory State: <i>IdTokenCached</i> for <Configured blocked IdToken fields>
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>Authorized</i> for <Configured blocked IdToken fields>	

Tool validations
N/a
Post scenario validations: - N/a

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

TC_C_11_CS: Authorization through authorization cache - Expired

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)	The Charging Station has an authorization cache AND the Charging Station supports at least one of the following local start authorization options: C-30, C-31, C-32, C-34 OR supports at least one of the following remote start authorization options: C-36, C-37

Before (Preparations)
Configuration State: AuthCacheCtrlr.AuthCacheEnabled is <i>true</i> (If implemented) AuthCtrlr.LocalPreAuthorize is <i>true</i> (If implemented) AuthCacheCtrlr.DisablePostAuthorize is <i>false</i> (If implemented)
Memory State: <i>IdTokenCached</i> for <Configured expired IdToken fields>
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>Authorized</i> for <Configured expired IdToken fields>	

Tool validations
N/a
Post scenario validations: - N/a

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

TC_C_12_CS: Authorization through authorization cache - Invalid & Accepted

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)	The Charging Station has an authorization cache AND the Charging Station supports at least one of the following local start authorization options: C-30, C-31, C-32, C-34 OR supports at least one of the following remote start authorization options: C-36, C-37

Before (Preparations)
Configuration State: AuthCacheCtrlr.AuthCacheEnabled is <i>true</i> (If implemented) AuthCtrlr.LocalPreAuthorize is <i>true</i> (If implemented) AuthCacheCtrlr.DisablePostAuthorize is <i>false</i> (If implemented)
Memory State: <i>IdTokenCached</i> for <Configured invalid IdToken fields>
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
<p>1. Execute Reusable State <i>Authorized</i> for <Configured invalid IdToken fields> (idTokenInfo.status Accepted) <u>Note:</u> <i>The configured invalid idToken fields are being used, however the Test system responds with status Accepted to simulate an outdated cache entry. In the mean time the idToken became valid.</i></p>	
<p>2. Execute Reusable State <i>EVConnectedPreSession</i></p>	
<p>3. Execute Reusable State <i>EnergyTransferStarted</i></p>	

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.

TC_C_13_CS: Authorization through authorization cache - Accepted but cable not connected yet.

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)	The Charging Station has an authorization cache AND the Charging Station supports at least one of the following local start authorization options: C-30, C-31, C-32, C-34 OR supports at least one of the following remote start authorization options: C-36, C-37

Before (Preparations)
Configuration State: AuthCacheCtrlr.AuthCacheEnabled is <i>true</i> (If implemented) AuthCtrlr.LocalPreAuthorize is <i>true</i> (If implemented)
Memory State: <i>IdTokenCached</i> for <Configured valid IdToken fields>
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>Authorized</i> (Cached idToken)	
2. Execute Reusable State <i>EVConnectedPreSession</i>	
3. Execute Reusable State <i>EnergyTransferStarted</i>	

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.

TC_C_14_CS: Authorization through authorization cache - GroupID equal to MasterPassGroupId.

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.Available 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 <i>true</i> (If implemented) AuthCtrlr.LocalPreAuthorize is <i>true</i> (If implemented) AuthCtrlr.MasterPassGroupId is <i><Configured MasterPassGroupId></i>
Memory State: Store the idToken that is part of the MasterPass group at the cache.
Reusable State(s): State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action</u> : Present the idToken with group id "MasterPassGroupId" which is already configured in the Authorization Cache	
1. The Charging Station sends a TransactionEventRequest	2. The Test System responds with a TransactionEventResponse
3. Execute Reusable State <i>EVConnectedPostSession</i>	
4. Execute Reusable State <i>EVDisconnected</i>	
5. Execute Reusable State <i>ParkingBayUnoccupied</i>	
<u>Note(s)</u> : Step 3, 4, and 5 are only executed if the transaction is still running.	

Tool validations
* Step 2: Message TransactionEventRequest - triggerReason <i>StopAuthorized</i> - idToken.idToken <i><Configured masterPass idToken></i> - idToken.type <i><Configured masterPass idTokenType></i> If eventType <i>Ended</i> , then: - transactionInfo.stoppedReason <i>MasterPass</i>
Post scenario validations: - N/a

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

TC_C_15_CS: Authorization through authorization cache - StopTxOnInvalidId = false, MaxEnergyOnInvalidId > 0

Test case name	Authorization through authorization cache - StopTxOnInvalidId = false, MaxEnergyOnInvalidId > 0
Test case Id	TC_C_15_CS
Use case Id(s)	C12
Requirement(s)	C12_FR_02, C12_FR_04
System under test	Charging Station
Description	This test case describes how the EV Driver is authorized to start a transaction while the Charging Station uses Cached IdToken. This enables the EV Driver to Online start a transaction by using the Authorization Cache in which the Charging Station can respond faster, as no AuthorizeRequest is being sent.
Purpose	To verify if the Charging Station is able to respond correctly to an idToken which has status "Accepted" in its cache but is "Invalid" in the CSMS with certain values of StopTxOnInvalidId and MaxEnergyOnInvalidId according to the mechanism as described in the OCPP specification.
Prerequisite(s)	<ul style="list-style-type: none"> - AuthCacheCtrlr.Available 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)
<p>Configuration State:</p> <p>AuthCacheCtrlr.AuthCacheEnabled is <i>true</i> (If implemented)</p> <p>AuthCtrlr.LocalPreAuthorize is <i>true</i> (If implemented)</p> <p>AuthCtrlr.LocalAuthorizeOffline is <i>true</i></p> <p>OfflineTxForUnknownIdEnabled is <i>true</i> (If implemented)</p> <p>StopTxOnInvalidId is <i>false</i></p> <p>MaxEnergyOnInvalidId is <i>10.000</i></p> <p>OfflineThreshold is <i><Configured RetryBackOffWaitMinimum_duration> + 60.0</i></p> <p>RetryBackOffWaitMinimum is <i><Configured RetryBackOffWaitMinimum_duration></i></p> <p>RetryBackOffRandomRange is <i>0</i></p> <p><u>Note:</u></p> <p><i><Configured RetryBackOffWaitMinimum_duration should be long enough to execute manual tasks></i></p> <p>Memory State:</p> <p><i>IdTokenCached</i> for <i><Configured valid IdToken fields></i> (If implemented) A known valid idToken is configured in the Local auth list (if implemented) and auth cache (if implemented)</p> <p>Reusable State(s):</p> <p>State is <i>EVConnectedPreSession</i></p>

Main (Test scenario)	
Charging Station	CSMS
1. The Test System closes the WebSocket connection AND does not accept a reconnect.	
<u>Manual Action</u> : Present valid idToken which is already configured in the Authorization Cache	
<u>Note(s)</u> : The Test System will wait for _<Configured Transaction Duration> seconds_	
2. The Test System accepts reconnection attempt from the Charging Station.	
<u>Note(s)</u> : - The Charging Station will empty its Transaction message queue. This will contain one or more TransactionEventRequest messages	
3. The Charging Station sends a TransactionEventRequest	4. The Test System responds with a TransactionEventResponse with idTokenInfo.status <i>Invalid</i> (if idToken is not omitted)

Tool validations
<p>* Step 3:</p> <p>Message TransactionEventRequest</p> <p>A message with (optional):</p> <ul style="list-style-type: none">- triggerReason <i>Authorized</i>- idToken.idToken <i><Configured valid_idtoken_idtoken></i>- idToken.type <i><Configured valid_idtoken_type></i>- offline <i>True</i> <p>A message with:</p> <ul style="list-style-type: none">- triggerReason <i>ChargingStateChanged</i>- offline <i>True</i> <p>No message with:</p> <ul style="list-style-type: none">- triggerReason <i>Deauthorized</i> or- triggerReason <i>ChargingStateChanged</i> and- transactionInfo.chargingState <i>SuspendedEVSE</i>
<p>Post scenario validations:</p> <ul style="list-style-type: none">- 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.

TC_C_16_CS: Authorization through authorization cache - StopTxOnInvalidId = true

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, C15.FR.04
System under test	Charging Station
Description	This test case describes how the EV Driver is authorized to start a transaction while the Charging Station uses Cached IdToken. This enables the EV Driver to Online start a transaction by using the Authorization Cache in which the Charging Station can respond faster, as no AuthorizeRequest is being sent.
Purpose	To verify if the Charging Station is able to respond correctly to an idToken which has status "Accepted" in its cache but is "Invalid" in the CSMS when StopTxOnInvalidId is true according to the mechanism as described in the OCPP specification.
Prerequisite(s)	<ul style="list-style-type: none"> - AuthCacheCtrlr.Available 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 Test System closes the WebSocket connection AND does not accept a reconnect.	
<u>Manual Action:</u> Present valid idToken which is already configured in the Authorization Cache	
<u>Note(s):</u> The Test System will wait for 5 seconds	
2. The Test System accepts reconnection attempt from the Charging Station.	
<u>Note(s):</u> - The Charging Station will empty its Transaction message queue. This will contain one or more TransactionEventRequest messages	
3. The Charging Station sends a TransactionEventRequest	4. The Test System responds with a TransactionEventResponse with idTokenInfo.status <i>Invalid</i> (if idToken is not omitted)
5. The Charging Station sends a TransactionEventRequest with triggerReason <i>Deauthorized</i>	6. The Test System responds with a TransactionEventResponse

Tool validations
<p>* Step 3:</p> <p>Message TransactionEventRequest</p> <p>A message with (optional):</p> <ul style="list-style-type: none">- triggerReason <i>Authorized</i>- idToken.idToken <i><Configured valid_idtoken_idtoken></i>- idToken.type <i><Configured valid_idtoken_type></i>- offline <i>True</i> <p>A message with:</p> <ul style="list-style-type: none">- triggerReason <i>ChargingStateChanged</i>- offline <i>True</i> <p>A message with:</p> <ul style="list-style-type: none">- triggerReason <i>Deauthorized</i>
<p>Post scenario validations:</p> <ul style="list-style-type: none">- Energyflow stops on receiving status invalid

NOTE

|

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

TC_C_17_CS: Authorization through authorization cache - StopTxOnInvalidId = false

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. C15.FR.06
System under test	Charging Station
Description	This test case describes how the EV Driver is authorized to start a transaction while the Charging Station uses Cached IdToken. This enables the EV Driver to Online start a transaction by using the Authorization Cache in which the Charging Station can respond faster, as no AuthorizeRequest is being sent.
Purpose	To verify if the Charging Station is able to respond correctly to an idToken which has status "Accepted" in its cache but is "Invalid" in the CSMS when StopTxOnInvalidId is false according to the mechanism as described in the OCPP specification.
Prerequisite(s)	<ul style="list-style-type: none"> - AuthCacheCtrlr.Available 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 <i>true</i> (If implemented) LocalPreAuthorize is <i>true</i> (If implemented) OfflineTxForUnknownIdEnabled is <i>true</i> (If implemented) StopTxOnInvalidId is <i>false</i> MaxEnergyOnInvalidId is <i>0</i>
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 <i>EVConnectedPreSession</i>

Main (Test scenario)	
Charging Station	CSMS
1. The Test System closes the WebSocket connection AND does not accept a reconnect.	
<u>Manual Action</u> : Present valid idToken which is already configured in the Authorization Cache	
<u>Note(s)</u> : The Test System will wait for 5 seconds	
2. The Test System accepts reconnection attempt from the Charging Station.	
<u>Note(s)</u> : - The Charging Station will empty its Transaction message queue. This will contain one or more TransactionEventRequest messages	
3. The Charging Station sends a TransactionEventRequest	4. The Test System responds with a TransactionEventResponse with idTokenInfo.status <i>Invalid</i> (if idToken is not omitted)
5. The Charging Station sends a TransactionEventRequest with triggerReason <i>SuspendedEVSE</i>	6. The Test System responds with a TransactionEventResponse

Tool validations
<p>* Step 2:</p> <p>Message TransactionEventRequest</p> <p>A message with:</p> <ul style="list-style-type: none">- triggerReason <i>Authorized</i>- idToken.idToken <i><Configured valid_idtoken_idtoken></i>- idToken.type <i><Configured valid_idtoken_type></i>- offline <i>True</i> <p>A message with:</p> <ul style="list-style-type: none">- transactionInfo.chargingState <i>SuspendedEVSE</i> <p>No message with: - triggerReason <i>SuspendedEVSE</i></p>
<p>Post scenario validations:</p> <ul style="list-style-type: none">- Energyflow stops on receiving status invalid

NOTE

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

TC_C_18_CS: Authorization through authorization cache - StopTxOnInvalidId = true, MaxEnergyOnInvalidId > 0

Test case name	Authorization through authorization cache - StopTxOnInvalidId = true, MaxEnergyOnInvalidId > 0
Test case Id	TC_C_18_CS
Use case Id(s)	C12
Requirement(s)	C12_FR_02, C12_FR_04
System under test	Charging Station
Description	This test case describes how the EV Driver is authorized to start a transaction while the Charging Station uses Cached IdToken. This enables the EV Driver to Online start a transaction by using the Authorization Cache in which the Charging Station can respond faster, as no AuthorizeRequest is being sent.
Purpose	To verify if the Charging Station is able to respond correctly to an idToken which has status "Accepted" in its cache but is "Invalid" in the CSMS when StopTxOnInvalidId is true and MaxEnergyOnInvalidId > 0 according to the mechanism as described in the OCPP specification.
Prerequisite(s)	<ul style="list-style-type: none"> - AuthCacheCtrlr.Available 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)
<p>Configuration State:</p> <p>AuthCacheEnabled is <i>true</i> (If implemented)</p> <p>LocalPreAuthorize is <i>true</i> (If implemented)</p> <p>LocalAuthorizeOffline is <i>true</i></p> <p>OfflineTxForUnknownIdEnabled is <i>true</i> (If implemented)</p> <p>StopTxOnInvalidId is <i>true</i></p> <p>MaxEnergyOnInvalidId is 500</p> <p>OfflineThreshold is <Configured RetryBackOffWaitMinimum_duration> + 60.0</p> <p>RetryBackOffWaitMinimum is <Configured RetryBackOffWaitMinimum_duration></p> <p>RetryBackOffRandomRange is 0</p> <p><u>Note:</u></p> <p><Configured RetryBackOffWaitMinimum_duration should be long enough to execute manual tasks></p> <p>Memory State:</p> <p><i>IdTokenCached</i> for <Configured valid IdToken fields> (If implemented)</p> <p>A known valid idToken is configured in the Local auth list (if implemented) and auth cache (if implemented)</p> <p>Reusable State(s):</p> <p>State is <i>EVConnectedPreSession</i></p>

Main (Test scenario)	
Charging Station	CSMS
1. The Test System closes the WebSocket connection AND does not accept a reconnect.	
<u>Manual Action</u> : Present valid idToken which is already configured in the Authorization Cache	
<u>Note(s)</u> : The Test System will wait for _<Configured Transaction Duration> seconds_	
2. The Test System accepts reconnection attempt from the Charging Station.	
<u>Note(s)</u> : - The Charging Station will empty its Transaction message queue. This will contain one or more TransactionEventRequest messages	
3. The Charging Station sends a TransactionEventRequest	4. The Test System responds with a TransactionEventResponse with idTokenInfo.status <i>Invalid</i> (if idToken is not omitted)
5. The Charging Station sends a TransactionEventRequest with triggerReason <i>Deauthorized</i>	6. The Test System responds with a TransactionEventResponse

Tool validations
<p>* Step 3:</p> <p>Message TransactionEventRequest</p> <p>A message with (optional):</p> <ul style="list-style-type: none">- triggerReason <i>Authorized</i>- idToken.idToken <i><Configured valid_idtoken_idtoken></i>- idToken.type <i><Configured valid_idtoken_type></i>- offline <i>True</i> <p>A message with:</p> <ul style="list-style-type: none">- triggerReason <i>ChargingStateChanged</i>- offline <i>True</i> <p>* Step 5:</p> <p>A message with:</p> <ul style="list-style-type: none">- triggerReason <i>Deauthorized</i>- offline <i>False</i>
<p>Post scenario validations:</p> <ul style="list-style-type: none">- Energyflow stops on receiving status invalid

NOTE

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

TC_C_57_CS: Authorization through authorization cache - AuthCacheDisablePostAuthorize

Test case name	Authorization through authorization cache - AuthCacheDisablePostAuthorize
Test case Id	TC_C_57_CS
Use case Id(s)	C12
Requirement(s)	C12.FR.05, C10.FR.03
System under test	Charging Station
Description	This test case describes how the EV Driver can be authorized to start a transaction by using Cached IdTokens. This enables the EV Driver to start a transaction while the Charging Station is online by using the Authorization Cache in which case the Charging Station can respond faster, since no AuthorizeRequest is being sent. For an IdToken that does not have status "Accepted" the Charging Station will still send an AuthorizeRequest to check against the most recent status from CSMS. However, when the setting AuthCacheDisablePostAuthorize is set to true, then the Charging Station will not do this.
Purpose	To verify that the Charging Station will not send an AuthorizeRequest for an IdToken in the Authorization Cache that is not "Accepted", when AuthCacheDisablePostAuthorize is set to true.
Prerequisite(s)	AuthCacheCtrlr.DisablePostAuthorize is implemented AND The Charging Station has an authorization cache AND the Charging Station supports at least one of the following local start authorization options: C-30, C-31, C-32, C-34 OR supports at least one of the following remote start authorization options: C-36, C-37

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): N/a

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action</u> : Present Invalid idToken which is already configured in the Authorization Cache	
<u>Note</u> : This step is executed if configured scenario is Local	
2. The Charging Station responds with a RequestStartTransactionResponse	1. The Test System sends a RequestStartTransactionRequest with idToken.idToken <Configured invalid_idtoken_idtoken> idToken.type <Configured invalid_idtoken_type> evseld <Configured evseld> <u>Note</u> : This step is executed if configured scenario is Remote
3. The Charging Station does NOT send a AuthorizeRequest	
4. Execute Reusable State <i>EVConnectedPreSession</i>	
5. The Charging Station does NOT start charging.	

Tool validations
* Step 3: Check that Charging Station does NOT send an AuthorizeRequest and authorization is refused.
Post scenario validations: The Charging Station does NOT start charging.

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

TC_C_27_CS: Online authorization through local authorization list - Accepted

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 AND - The Charging Station must support an authorization method other than NoAuthorization or Central

Before (Preparations)
Configuration State: LocalAuthListEnabled is <i>true</i> (If implemented) LocalPreAuthorize is <i>true</i> (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
1. Execute Reusable State <i>Authorized</i> (Stored idToken)	
2. Execute Reusable State <i>EVConnectedPreSession</i>	
3. Execute Reusable State <i>EnergyTransferStarted</i>	

Tool validations
N/a
Post scenario validations: - Energy is transferred

TC_C_28_CS: Online authorization through local authorization list - Invalid & Not Accepted

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 AND - The Charging Station must support an authorization method other than NoAuthorization or Central

Before (Preparations)
Configuration State: LocalAuthListEnabled is <i>true</i> (If implemented) LocalPreAuthorize is <i>true</i> (If implemented) LocalAuthListDisablePostAuthorize <i>false</i> (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 <i>Authorized</i> (idTokenInfo.status Invalid)	

Tool validations
N/a
Post scenario validations: - N/a

TC_C_29_CS: Online authorization through local authorization list - Blocked

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)	<ul style="list-style-type: none"> - LocalAuthListCtrlr.LocalAuthListAvailable is implemented with value true AND - The Charging Station must support an authorization method other than NoAuthorization or Central

Before (Preparations)
Configuration State: LocalAuthListEnabled is <i>true</i> (If implemented) LocalPreAuthorize is <i>true</i> (If implemented) <i>*LocalAuthListDisablePostAuthorize * false</i> (If implemented)
Memory State: A known blocked idToken is configured in the Local Authorization List
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State Authorized (idTokenInfo.status Blocked)	

Tool validations
N/a
Post scenario validations: - N/a

TC_C_30_CS: Online authorization through local authorization list - Expired

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)	<ul style="list-style-type: none"> - LocalAuthListCtrlr.LocalAuthListAvailable is implemented with value true AND - The Charging Station must support an authorization method other than NoAuthorization or Central

Before (Preparations)
Configuration State: LocalAuthListEnabled is <i>true</i> (If implemented) LocalPreAuthorize is <i>true</i> (If implemented) <i>*LocalAuthListDisablePostAuthorize * false</i> (If implemented)
Memory State: A known expired idToken is configured in the Local Authorization List
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>Authorized</i> (idTokenInfo.status Expired)	

Tool validations
N/a
Post scenario validations: - N/a

TC_C_31_CS: Online authorization through local authorization list - Invalid & Accepted

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 AND - The Charging Station must support an authorization method other than NoAuthorization or Central

Before (Preparations)
Configuration State: LocalAuthListEnabled is <i>true</i> (If implemented) LocalPreAuthorize is <i>true</i> (If implemented) *LocalAuthListDisablePostAuthorize * <i>false</i> (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
<p>1. Execute Reusable State <i>Authorized</i> for <Configured invalid IdToken fields> (idTokenInfo.status Accepted) <i>Note: The configured invalid idToken fields are being used, however the Test system responds with status Accepted to simulate an outdated local authorization list entry. In the mean time the idToken became valid.</i></p>	
<p>2. Execute Reusable State <i>EVConnectedPreSession</i></p>	
<p>3. Execute Reusable State <i>EnergyTransferStarted</i></p>	

Tool validations
- N/a
Post scenario validations: - Energy is transferred

TC_C_58_CS: Online authorization through local authorization list - LocalAuthListDisablePostAuthorize

Test case name	Online authorization through local authorization list - LocalAuthListDisablePostAuthorize
Test case Id	TC_C_58_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)	<ul style="list-style-type: none"> - LocalAuthListCtrlr.Available is implemented with value <i>true</i> AND - LocalAuthListCtrlr.DisablePostAuthorize is implemented AND - The Charging Station must support an authorization method other than NoAuthorization or Central

Before (Preparations)
Configuration State: LocalAuthListCtrlr.Enabled is <i>true</i> (If implemented) AuthCtrlr.LocalPreAuthorize is <i>true</i> (If implemented) LocalAuthListCtrlr.DisablePostAuthorize <i>true</i>
Memory State: <i>IdTokenLocalAuthList</i> for <Configured invalid idtoken fields>
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action</u> : Present Invalid idToken which is already stored in the Local Authorization List.	
<u>Note</u> : This step is executed if configured scenario is Local	
2. The Charging Station responds with a RequestStartTransactionResponse	1. The Test System sends a RequestStartTransactionRequest with idToken.idToken <Configured invalid_idtoken_idtoken> idToken.type <Configured invalid_idtoken_type> evseld <Configured evseld> <u>Note</u> : This step is executed if configured scenario is Remote
3. The Charging Station does NOT send a AuthorizeRequest	
4. Execute Reusable State <i>EVConnectedPreSession</i>	
5. The Charging Station does NOT start charging.	

Tool validations
* Step 3: Check that Charging Station does NOT send an AuthorizeRequest and authorization is refused.
Post scenario validations: The Charging Station does NOT start charging.

TC_C_32_CS: Store Authorization Data in the Authorization Cache - Persistent over reboot

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 has an authorization cache AND Authorization cache is stored in the non-volatile memory AND the Charging Station supports at least one of the following local start authorization options: C-30, C-31, C-32, C-34 OR supports at least one of the following remote start authorization options: C-36, C-37

Before (Preparations)
Configuration State: AuthCacheEnabled is <i>true</i> (If implemented) LocalPreAuthorize is <i>true</i>
Memory State: <i>IdTokenCached</i> for <Configured valid IdToken fields>
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>Booted</i>	
2. Execute Reusable State <i>Authorized</i> (Cached idToken)	
3. Execute Reusable State <i>EVConnectedPreSession</i>	
4. Execute Reusable State <i>EnergyTransferStarted</i>	

Tool validations
N/a
Post scenario validations: - N/a

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

TC_C_33_CS: Store Authorization Data in the Authorization Cache - Update on AuthorizeResponse

Test case name	Store Authorization Data in the Authorization Cache - Update on AuthorizeResponse
Test case Id	TC_C_33_CS
Use case Id(s)	C10
Requirement(s)	C10.FR.04, C12.FR.06
System under test	Charging Station
Description	This test case covers how the Charging Station autonomously stores a record of previously presented identifiers that have been successfully authorized by the CSMS in the Authorization Cache. (Successfully meaning: a response received on a message containing an IdToken)
Purpose	To verify if the Charging Station is able to store the identifiers correctly upon an AuthorizeResponse according to the mechanism as described in the OCPP specification.
Prerequisite(s)	The Charging Station has an authorization cache AND the Charging Station supports at least one of the following local start authorization options: C-30, C-31, C-32, C-34 OR supports at least one of the following remote start authorization options: C-36, C-37

Before (Preparations)
Configuration State: AuthCacheEnabled is <i>true</i> (If implemented) LocalPreAuthorize is <i>true</i> LocalAuthListEnabled is <i>true</i>
Memory State: N/a
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>IdTokenCached</i> for <Configured valid IdToken fields>	
2. Execute Reusable State <i>Authorized</i> (Cached idToken)	
3. Execute Reusable State <i>EVConnectedPreSession</i>	
4. Execute Reusable State <i>EnergyTransferStarted</i>	

Tool validations
N/a
Post scenario validations: - N/a

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

TC_C_34_CS: Store Authorization Data in the Authorization Cache - Update on TransactionResponse

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)	The Charging Station has an authorization cache AND the Charging Station supports at least one of the following local start authorization options: C-30, C-31, C-32, C-34 OR supports at least one of the following remote start authorization options: C-36, C-37

Before (Preparations)
Configuration State: AuthCacheEnabled is true (If implemented) LocalPreAuthorize is true LocalAuthListEnabled is true StopTxOnInvalidId is true MaxEnergyOnInvalidId is 0
Memory State: IdTokenCached for <Configured valid IdToken fields>
Reusable State(s): N/a

Main (Test scenario)		
Charging Station		CSMS
1. Execute Reusable State <i>Authorized</i> (Cached idToken, idTokenInfo.status invalid)		
2. Execute Reusable State <i>Deauthorized</i>		
3. Execute Reusable State <i>EVDisconnected</i>		
4. Execute Reusable State <i>ParkingBayUnoccupied</i>		
5. Execute Reusable State <i>ParkingBayOccupied</i>		
6. Execute Reusable State <i>Authorized</i> (idTokenInfo.status invalid)		

Tool validations
N/a
Post scenario validations: - N/a

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

TC_C_36_CS: Store Authorization Data in the Authorization Cache - AuthCacheCtrlr.LocalPreAuthorize = false

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)	The Charging Station has an authorization cache AND the Charging Station supports at least one of the following local start authorization options: C-30, C-31, C-32, C-34 OR supports at least one of the following remote start authorization options: C-36, C-37

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
1. Execute Reusable State <i>Authorized</i> (idTokenInfo.status invalid)	

Tool validations
N/a
Post scenario validations: - N/a

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

TC_C_37_CS: Clear Authorization Data in Authorization Cache - Accepted

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)	The Charging Station has an authorization cache AND the Charging Station supports at least one of the following local start authorization options: C-30, C-31, C-32, C-34 OR supports at least one of the following remote start authorization options: C-36, C-37

Before (Preparations)
Configuration State: AuthCacheEnabled is <i>true</i> (If implemented)
Memory State: <i>IdTokenCached</i> for <Configured valid IdToken fields>
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
	1. The Test System sends a ClearCacheRequest
2. The Charging Station responds with a ClearCacheResponse	
3. Execute Reusable State <i>Authorized</i>	
4. Execute Reusable State <i>EVConnectedPreSession</i>	
5. Execute Reusable State <i>EnergyTransferStarted</i>	

Tool validations
* Step 2: Message ClearCacheResponse - status <i>Accepted</i>
Post scenario validations: - N/a

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

TC_C_38_CS: Clear Authorization Data in Authorization Cache - Rejected

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)	The Charging Station has an authorization cache AND the Charging Station supports at least one of the following local start authorization options: C-30, C-31, C-32, C-34 OR supports at least one of the following remote start authorization options: C-36, C-37

Before (Preparations)
Configuration State: AuthCacheEnabled is <i>false</i> (If implemented)
Memory State: <i>IdTokenCached</i> 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 Test System sends a ClearCacheRequest

Tool validations
* Step 2: Message ClearCacheResponse - status <i>Rejected</i>
Post scenario validations: - N/a

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

TC_C_39_CS: Authorization by GroupId - Success

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 <i>EVConnectedPreSession</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action:</u> Present valid idToken with <Configured GroupId>	
1. The Charging Station sends an AuthorizeRequest	2. The Test System responds with an AuthorizeResponse with - idTokenInfo.status <i>Accepted</i> - idTokenInfo.groupIdToken.idToken <Configured groupIdToken>
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 Test System responds with a TransactionEventResponse with - idTokenInfo.status <i>Accepted</i> - idTokenInfo.groupIdToken.idToken <Configured groupIdToken>
5. Execute Reusable State <i>EnergyTransferStarted</i>	
<u>Manual Action:</u> Present other valid idToken with <Configured GroupId>	
6. The Charging Station sends an AuthorizeRequest	7. The Test System responds with an AuthorizeResponse with - idTokenInfo.status <i>Accepted</i> - idTokenInfo.groupIdToken.idToken <Configured groupIdToken>
8. The Charging Station sends a TransactionEventRequest	9. The Test System responds with a TransactionEventResponse with - idTokenInfo.status <i>Accepted</i> - idTokenInfo.groupIdToken.idToken <Configured groupIdToken>
10. Execute Reusable State <i>EVConnectedPostSession</i>	
11. Execute Reusable State <i>EVDDisconnected</i>	
12. Execute Reusable State <i>ParkingBayUnoccupied</i>	

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

TC_C_40_CS: Authorization by GroupId - Success with Local Authorization List

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)	<ul style="list-style-type: none"> - 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
<u>Manual Action:</u> Present valid idToken with <Configured GroupId> which is configured in the Local Authorization List	
1. The Charging Station sends a TransactionEventRequest <u>Note(s):</u> - This step needs to be executed when TxStartPoint contains Authorized OR the transaction already started. So in the case TxStartPoint contains ParkingBayOccupancy	2. The Test System responds with a TransactionEventResponse with - idTokenInfo.status Accepted - idTokenInfo.groupIdToken.idToken <Configured groupIdToken>
3. Execute Reusable State <i>EnergyTransferStarted</i>	
<u>Manual Action:</u> Present valid idToken2 with <Configured GroupId> which is configured in the Authorization Cache	
4. Execute Reusable State <i>StopAuthorized</i>	
5. Execute Reusable State <i>EVConnectedPostSession</i>	
6. Execute Reusable State <i>EVDisconnected</i>	
7. Execute Reusable State <i>ParkingBayUnoccupied</i>	

Tool validations
<p>* Step 1:</p> <p>Message TransactionEventRequest</p> <ul style="list-style-type: none">- triggerReason <i>Authorized</i>- idToken.idToken <i><Configured valid_idtoken_idtoken></i>- idToken.type <i><Configured valid_idtoken_type></i> <p>if transaction was already started</p> <ul style="list-style-type: none">- eventType <i>Updated</i> <p>else</p> <ul style="list-style-type: none">- eventType <i>Started</i> <p>* Step 4:</p> <p>Message TransactionEventRequest</p> <ul style="list-style-type: none">- triggerReason <i>StopAuthorized</i>- idToken.idToken <i><Configured valid_idtoken_idtoken></i>- idToken.type <i><Configured valid_idtoken_type></i>- eventType <i>Updated</i>
<p>Post scenario validations:</p> <ul style="list-style-type: none">- N/a

TC_C_41_CS: Authorization by GroupId - Success with Authorization Cache

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)	<ul style="list-style-type: none"> - AuthCacheCtrlr.Available 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
<u>Manual Action:</u> Present valid idToken with <Configured GroupId> which is configured in the Authorization Cache	
1. The Charging Station sends a TransactionEventRequest <u>Note(s):</u> - This step needs to be executed when TxStartPoint contains Authorized OR the transaction already started. So in the case TxStartPoint contains ParkingBayOccupancy	2. The Test System responds with a TransactionEventResponse with - idTokenInfo.status Accepted - idTokenInfo.groupIdToken.idToken <Configured groupIdToken>
3. Execute Reusable State <i>EnergyTransferStarted</i>	
<u>Manual Action:</u> Present valid idToken2 with <Configured GroupId> which is configured in the Authorization Cache	
4. Execute Reusable State <i>StopAuthorized</i>	
5. Execute Reusable State <i>EVConnectedPostSession</i>	
6. Execute Reusable State <i>EVDisconnected</i>	
7. Execute Reusable State <i>ParkingBayUnoccupied</i>	

Tool validations
<div>* Step 1: Message TransactionEventRequest - triggerReason <i>Authorized</i> - idToken.idToken <i><Configured valid_idtoken_idtoken></i> - idToken.type <i><Configured valid_idtoken_type></i> if transaction was already started - eventType <i>Updated</i> else - eventType <i>Started</i> * Step 4: Message TransactionEventRequest - triggerReason <i>StopAuthorized</i> - idToken.idToken <i><Configured valid_idtoken_idtoken></i> - idToken.type <i><Configured valid_idtoken_type></i></div>
<div>Post scenario validations: - N/a</div>

NOTE

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

TC_C_42_CS: Authorization by GroupId - Not stopped by GroupId

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
<u>Manual Action:</u> Present valid idToken with <Configured GroupId>	
1. The Charging Station sends an AuthorizeRequest	2. The Test System responds with an AuthorizeResponse with - idTokenInfo.status <i>Accepted</i> - idTokenInfo.groupIdToken.idToken <Configured groupIdToken>
3. The Charging Station sends a TransactionEventRequest <u>Note(s):</u> - This step needs to be executed when TxStartPoint contains <i>Authorized</i> OR the transaction already started. So in the case TxStartPoint contains <i>ParkingBayOccupancy</i>	4. The Test System responds with a TransactionEventResponse with - idTokenInfo.status <i>Accepted</i> - idTokenInfo.groupIdToken.idToken <Configured groupIdToken>
5. Execute Reusable State <i>EnergyTransferStarted</i>	
<u>Manual Action:</u> Present invalid idToken with <Configured GroupId>	
6. The Charging Station sends an AuthorizeRequest	7. The Test System responds with an AuthorizeResponse with - idTokenInfo.status <i>Invalid</i> - idTokenInfo.groupIdToken.idToken <Configured groupIdToken>
<u>Note(s):</u> Test System will wait to see if CS indeed doesn't send a <i>TransactionEventRequest</i>	

Tool validations
<p>* Step 1:</p> <p>Message AuthorizeRequest</p> <ul style="list-style-type: none">- idToken.idToken <Configured valid_idtoken_idtoken>- idToken.type <Configured valid_idtoken_type> <p>* Step 3:</p> <p>Message TransactionEventRequest</p> <ul style="list-style-type: none">- triggerReason <i>Authorized</i>- idToken.idToken <Configured valid_idtoken_idtoken>- idToken.type <Configured valid_idtoken_type> <p>if transaction was already started</p> <ul style="list-style-type: none">- eventType <i>Updated</i> <p>else</p> <ul style="list-style-type: none">- eventType <i>Started</i> <p>* Step 6:</p> <p>Message AuthorizeRequest</p> <ul style="list-style-type: none">- idToken.idToken <Configured valid_idtoken_idtoken>- idToken.type <Configured valid_idtoken_type>
<p>Post scenario validations:</p> <ul style="list-style-type: none">- The energy transfer is not stopped

TC_C_43_CS: Authorization by GroupId - Invalid status with Local Authorization List

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)	<ul style="list-style-type: none"> - LocalAuthListCtrlr.LocalAuthListAvailable is implemented with value true - The Charging Station supports authorization methods other than NoAuthorization

Before (Preparations)
Configuration State: LocalAuthListEnabled is <i>true</i> (If implemented) LocalPreAuthorize is <i>true</i> (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 <i>EVConnectedPreSession</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action:</u> Present valid idToken with <Configured GroupId> which is configured in the Local Authorization List	
1. The Charging Station sends a TransactionEventRequest <u>Note(s):</u> - This step needs to be executed when TxStartPoint contains Authorized OR the transaction already started. So in the case TxStartPoint contains ParkingBayOccupancy	2. The Test System responds with a TransactionEventResponse with - idTokenInfo.status Accepted - idTokenInfo.groupIdToken.idToken <Configured groupIdToken>
3. Execute Reusable State <i>EnergyTransferStarted</i>	
<u>Manual Action:</u> Present invalid idToken with <Configured GroupId> which is configured in the Local Authorization List	
4. The Charging Station sends an AuthorizeRequest	5. The Test System responds with an AuthorizeResponse with - idTokenInfo.status Invalid - idTokenInfo.groupIdToken.idToken <Configured groupIdToken>
<u>Note(s):</u> Test System will wait to see if CS indeed doesn't send a TransactionEventRequest	

Tool validations
<p>* Step 1:</p> <p>Message TransactionEventRequest</p> <ul style="list-style-type: none">- triggerReason <i>Authorized</i>- idToken.idToken <i><Configured valid_idtoken_idtoken></i>- idToken.type <i><Configured valid_idtoken_type></i> <p>if transaction was already started</p> <ul style="list-style-type: none">- eventType <i>Updated</i> <p>else</p> <ul style="list-style-type: none">- eventType <i>Started</i> <p>* Step 4:</p> <p>Message AuthorizeRequest</p> <ul style="list-style-type: none">- idToken.idToken <i><Configured valid_idtoken_idtoken></i>- idToken.type <i><Configured valid_idtoken_type></i> <p>* Step 6:</p> <p>Message TransactionEventRequest</p> <ul style="list-style-type: none">- triggerReason <i>StopAuthorized</i>- idToken.idToken <i><Configured valid_idtoken_idtoken></i>- idToken.type <i><Configured valid_idtoken_type></i>- eventType <i>Updated</i>
<p>Post scenario validations:</p> <ul style="list-style-type: none">- N/a

TC_C_44_CS: Authorization by GroupId - Invalid status with Authorization Cache

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)	<ul style="list-style-type: none"> - AuthCacheCtrlr.Available 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
<u>Manual Action:</u> Present valid idToken with <Configured GroupId> which is configured in the Authorization Cache	
1. The Charging Station sends a TransactionEventRequest <u>Note(s):</u> - This step needs to be executed when TxStartPoint contains Authorized OR the transaction already started. So in the case TxStartPoint contains ParkingBayOccupancy	2. The Test System responds with a TransactionEventResponse with - idTokenInfo.status Accepted - idTokenInfo.groupIdToken.idToken <Configured groupIdToken>
3. Execute Reusable State <i>EnergyTransferStarted</i> <u>Manual Action:</u> Present invalid idToken with <Configured GroupId> which is configured in the Authorization Cache	
4. The Charging Station sends an AuthorizeRequest	5. The Test System responds with an AuthorizeResponse with - idTokenInfo.status Invalid - idTokenInfo.groupIdToken.idToken <Configured groupIdToken>
<u>Note(s):</u> Test System will wait to see if CS indeed doesn't send a TransactionEventRequest	

Tool validations
<div>* Step 1: Message TransactionEventRequest - triggerReason <i>Authorized</i> - idToken.idToken <i><Configured valid_idtoken_idtoken></i> - idToken.type <i><Configured valid_idtoken_type></i> if transaction was already started - eventType <i>Updated</i> else - eventType <i>Started</i> * Step 4: Message AuthorizeRequest - idToken.idToken <i><Configured invalid_idtoken_idtoken></i> - idToken.type <i><Configured invalid_idtoken_type></i></div>
<div>Post scenario validations: - N/a</div>

NOTE

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

TC_C_45_CS: Authorization by GroupId - Master pass - Not able to start transaction + groupId

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)	<ul style="list-style-type: none"> - The Charging station supports MasterPass feature. - The Charging Station supports authorization methods other than NoAuthorization

Before (Preparations)
Configuration State: TxCtrlr.TxStartPoint should contain <i>Authorized</i> or <i>PowerPathClosed</i> and not contain <i>ParkingBayOccupancy</i> or <i>EVConnected</i> AuthCtrlr.MasterPassGroupId is <Configured MasterPassGroupId>
Memory State: N/a
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action:</u> <i>Present configured masterpass idToken</i>	
1. The Charging Station sends an AuthorizeRequest	2. The Test System responds with an AuthorizeResponse with - idTokenInfo.status <i>Accepted</i> - idTokenInfo.groupIdToken.idToken <i><Configured masterPassGroupId></i>
<u>Note:</u> <i>The Charging Station will not authorize the transaction and send a TransactionEventRequest (in case of TxStartPoint Authorized).</i>	
3. Execute Reusable State <i>EVConnectedPreSession</i>	
4. The Charging Station will NOT send a TransactionEventRequest with chargingState <i>Charging</i> and triggerReason <i>ChargingStateChanged</i>	

Tool validations
* Step 1: Message AuthorizeRequest - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type>
Post scenario validations: - N/a

TC_C_46_CS: Store Authorization Data in the Authorization Cache - AuthCacheLifeTime

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.Available is implemented with value true - Configuration variable AuthCacheLifeTime is implemented

Before (Preparations)
Configuration State: AuthCacheLifeTime is <Configured TransactionDuration> AuthCacheCtrlr.LocalPreAuthorize is true (If implemented)
Memory State: IdTokenCached <Configured valid idtoken fields>
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
1. Wait for <Configured Transaction Duration> seconds	
2. Execute Reusable State Authorized	

Tool validations
N/a
Post scenario validations: - N/a

TC_C_47_CS: Stop Transaction with a Master Pass - With UI - All transactions

Test case name	Stop Transaction with a Master Pass - With UI - All transactions
Test case Id	TC_C_47_CS
Use case Id(s)	C16
Requirement(s)	C16_FR_01
System under test	Charging Station
Description	This test case covers how somebody with a Master Pass (User) can stop (selected) ongoing transactions, so the cable becomes unlocked. This Master Pass can be configured in: MasterPassGroupId. This could for example be usefull for Law Enforcement officials.
Purpose	To verify if the Charging Station is able to correctly stop all transactions when an idToken which has the MasterPass as GroupId is used and the user has selected to stop all transactions in the User Interface according to the mechanism as described in the OCPP specification.
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station supports at least one authorization method described at the following Use cases; C01, C04. - Charging station has a User Interface - 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 <i>EnergyTransferStarted</i> for all EVSE

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action:</u> Present configured masterpass idToken	
1. The Charging Station sends an AuthorizeRequest	2. The Test System responds with an AuthorizeResponse with idTokenInfo.status <i>Accepted</i> idTokenInfo.groupIdToken.idToken <i><Configured masterPassGroupId></i>
<u>Manual Action:</u> Select to stop all transactions	
3. The Charging Station sends a TransactionEventRequest for all EVSE	4. The Test System responds with a TransactionEventResponse with idTokenInfo.status <i>Accepted</i> idTokenInfo.groupIdToken.idToken <i><Configured masterPassGroupId></i> for all EVSE
<u>Note(s):</u> The charging Station will repeat for all running transactions and can stop a transaction in more than one TransactionEventRequest.	
5. Execute Reusable State <i>EVConnectedPostSession</i> for all EVSE	
6. Execute Reusable State <i>EVDisconnected</i> for all EVSE	
7. Execute Reusable State <i>ParkingBayUnoccupied</i> for all EVSE	

Tool validations
<p>* Step 1:</p> <p>Message AuthorizeRequest</p> <ul style="list-style-type: none">- idToken.idToken <Configured masterpass_idtoken_idtoken>- idToken.type <Configured masterpass_idtoken_type> <p>* Step 3:</p> <p>Message TransactionEventRequest</p> <ul style="list-style-type: none">- transactionInfo.stoppedReason <i>MasterPass</i> (in last TransactionEventRequest)- idToken <i>omit</i> or- idToken.idToken <Configured masterpass_idtoken_idtoken> and- idToken.type <Configured masterpass_idtoken_type> (once per stopped transaction)- eventType <i>Ended</i> (in last TransactionEventRequest)
<p>Post scenario validations:</p> <ul style="list-style-type: none">- N/a

TC_C_48_CS: Stop Transaction with a Master Pass - With UI - With UI - Specific transactions

Test case name	Stop Transaction with a Master Pass - With UI - With UI - Specific transactions
Test case Id	TC_C_48_CS
Use case Id(s)	C16
Requirement(s)	C16_FR_01
System under test	Charging Station
Description	This test case covers how somebody with a Master Pass (User) can stop (selected) ongoing transactions, so the cable becomes unlocked. This Master Pass can be configured in: MasterPassGroupId. This could for example be usefull for Law Enforcement officials.
Purpose	To verify if the Charging Station is able to correctly stop a transaction when an idToken which has the MasterPass as GroupId is used and the user has selected to stop one transaction in the User Interface according to the mechanism as described in the OCPP specification.
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station supports at least one authorization method described at the following Use cases; C01, C04. - Charging station has a User Interface - 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 <i>EnergyTransferStarted</i> for all EVSE

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action:</u> Present configured masterpass idToken	
1. The Charging Station sends an AuthorizeRequest	2. The Test System responds with an AuthorizeResponse with idTokenInfo.status <i>Accepted</i> idTokenInfo.groupIdToken.idToken <Configured masterPassGroupId>
<u>Manual Action:</u> Select to stop the transaction on EVSE 1	
3. The Charging Station sends a TransactionEventRequest	4. The Test System responds with a TransactionEventResponse with idTokenInfo.status <i>Accepted</i> idTokenInfo.groupIdToken.idToken <Configured masterPassGroupId>
5. Execute Reusable State <i>EVConnectedPostSession</i>	
6. Execute Reusable State <i>EVDisconnected</i>	
7. Execute Reusable State <i>ParkingBayUnoccupied</i>	

Tool validations
<p>* Step 1:</p> <p>Message AuthorizeRequest</p> <ul style="list-style-type: none">- idToken.idToken <Configured masterpass_idtoken_idtoken>- idToken.type <Configured masterpass_idtoken_type> <p>* Step 3:</p> <p>Message TransactionEventRequest</p> <ul style="list-style-type: none">- transactionInfo.stoppedReason <i>MasterPass</i>- idToken <i>omit</i> or- idToken.idToken <Configured masterpass_idtoken_idtoken> and- idToken.type <Configured masterpass_idtoken_type>
<p>Post scenario validations:</p> <ul style="list-style-type: none">- All other EVSE still transfer energy

TC_C_49_CS: Stop Transaction with a Master Pass - Without UI

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)	<ul style="list-style-type: none"> - The Charging Station supports at least one authorization method described at the following Use cases; C01, C04. - The Charging Station 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
<u>Manual Action:</u> Present configured masterpass idToken	
1. The Charging Station sends an AuthorizeRequest	2. The Test System responds with an AuthorizeResponse with idTokenInfo.status <i>Accepted</i> idTokenInfo.groupIdToken.idToken <Configured masterPassGroupId>
3. The Charging Station sends a TransactionEventRequest for EVSE 1 (and 2)	4. The Test System responds with a TransactionEventResponse with idTokenInfo.status <i>Accepted</i> idTokenInfo.groupIdToken.idToken <Configured masterPassGroupId> for EVSE 1 (and 2)
5. Execute Reusable State <i>EVConnectedPostSession</i> for EVSE 1 (and 2)	
6. Execute Reusable State <i>EVDisconnected</i> for EVSE 1 (and 2)	
7. Execute Reusable State <i>ParkingBayUnoccupied</i> for EVSE 1 (and 2)	

Tool validations
<p>* Step 1:</p> <p>Message AuthorizeRequest</p> <ul style="list-style-type: none">- idToken.idToken <Configured masterpass_idtoken_idtoken>- idToken.type <Configured masterpass_idtoken_type> <p>* Step 3:</p> <p>Message TransactionEventRequest</p> <ul style="list-style-type: none">- transactionInfo.stoppedReason <i>MasterPass</i>- idToken <i>omit</i> or- idToken.idToken <Configured masterpass_idtoken_idtoken> and- idToken.type <Configured masterpass_idtoken_type>
<p>Post scenario validations:</p> <ul style="list-style-type: none">- N/a

TC_C_21_CS: Offline authorization through local authorization list - Accepted

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)	<ul style="list-style-type: none"> - LocalAuthListCtrlr.LocalAuthListAvailable is implemented with value true - The Charging Station supports authorization methods other than NoAuthorization

Before (Preparations)
Configuration State: LocalAuthListEnabled is <i>true</i> (If implemented) LocalPreAuthorize is <i>true</i> (If implemented) OfflineTxForUnknownIdEnabled is <i>false</i> (If implemented) LocalAuthorizeOffline is <i>true</i>
Memory State: <i>IdTokenLocalAuthList</i> for <Configured valid idtoken fields>
Reusable State(s): State is <i>StartOfflineTransaction</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action</u> : Present idToken.	
<u>Manual Action</u> : Unplug cable.	
<u>Manual Action</u> : Drive out of parkingbay.	
1. The Charging Stations sends a TransactionEventRequest <u>Note(s)</u> : - The Charging Station will empty its Transaction message queue. This will contain one or more TransactionEventRequest messages	2. The Test System responds with a TransactionEventResponse <u>Note(s)</u> : - The Test System will respond to the TransactionEventRequest containing the idToken, with idtokenInfo.status Accepted
3. Execute Reusable State <i>ParkingBayUnoccupied</i>	

Tool validations
<p>* Step 1:</p> <p>Message(s) before the StopAuthorize: TransactionEventRequests</p> <ul style="list-style-type: none"> - offline must be <i>true</i> <p>One of the Message(s): TransactionEventRequest</p> <ul style="list-style-type: none"> - TriggerReason must be <i>StopAuthorized</i>
Post scenario validations: N/a

TC_C_22_CS: Offline authorization through local authorization list - Invalid

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)	<ul style="list-style-type: none"> - LocalAuthListCtrlr.LocalAuthListAvailable is implemented with value true - The Charging Station supports authorization methods other than NoAuthorization

Before (Preparations)
<p>Configuration State:</p> <p>LocalAuthListEnabled is <i>true</i> (If implemented)</p> <p>LocalPreAuthorize is <i>true</i> (If implemented)</p> <p>OfflineTxForUnknownIdEnabled is <i>false</i> (If implemented)</p> <p>LocalAuthorizeOffline is <i>true</i></p> <p>OfflineThreshold is <i><Configured RetryBackOffWaitMinimum_duration> + 60.0</i></p> <p>RetryBackOffWaitMinimum is <i><Configured RetryBackOffWaitMinimum_duration></i></p> <p>RetryBackOffRandomRange is <i>0</i></p> <p>Memory State:</p> <p><i>IdTokenLocalAuthList</i> for <i><Configured invalid idtoken fields></i></p> <p>Reusable State(s):</p> <p>N/a</p>

Main (Test scenario)	
Charging Station	CSMS
1. <i>The Test System closes the WebSocket connection AND does not accept a reconnect.</i>	
2. <u>Manual Action</u> : <i>Drive EV into parking bay.</i>	
3. <u>Manual Action</u> : <i>Present idToken.</i>	
4. <i>The Test System accepts reconnection attempt from the Charging Station.</i>	
5. The Charging Stations sends a TransactionEventRequest	6. The Test System responds with a TransactionEventResponse
<u>Note(s)</u> : - <i>The Charging Station will empty its Transaction message queue, this will only contain TransactionEventRequests if TxStartPoint was ParkingBayOccupancy and a Transaction was started.</i>	

Tool validations
<p>* Step 5:</p> <p>Message TransactionEventRequest</p> <ul style="list-style-type: none"> - offline must be <i>true</i> - TriggerReason must be <i>EVDetected</i> <p>Post scenario validations:</p> <p>N/a</p>

TC_C_23_CS: Offline authorization through local authorization list - Blocked

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)	<ul style="list-style-type: none"> - LocalAuthListCtrlr.LocalAuthListAvailable is implemented with value true - The Charging Station supports authorization methods other than NoAuthorization

Before (Preparations)
<p>Configuration State:</p> <p>LocalAuthListEnabled is <i>true</i> (If implemented)</p> <p>LocalPreAuthorize is <i>true</i> (If implemented)</p> <p>OfflineTxForUnknownIdEnabled is <i>false</i> (If implemented)</p> <p>LocalAuthorizeOffline is <i>true</i></p> <p>OfflineThreshold is <i><Configured RetryBackOffWaitMinimum_duration> + 60.0</i></p> <p>RetryBackOffWaitMinimum is <i><Configured RetryBackOffWaitMinimum_duration></i></p> <p>RetryBackOffRandomRange is <i>0</i></p> <p>Memory State:</p> <p><i>IdTokenLocalAuthList</i> for <i><Configured blocked idtoken fields></i></p> <p>Reusable State(s):</p> <p>N/a</p>

Main (Test scenario)	
Charging Station	CSMS
1. The Test System closes the WebSocket connection AND does not accept a reconnect.	
2. <u>Manual Action</u> : Drive EV into parking bay.	
3. <u>Manual Action</u> : Present idToken.	
4. The Test System accepts reconnection attempt from the Charging Station.	
5. The Charging Stations sends a TransactionEventRequest	6. The Test System responds with a TransactionEventResponse
<u>Note(s)</u> : - The Charging Station will empty its Transaction message queue, this will only contain TransactionEventRequests if TxStartPoint was ParkingBayOccupancy and a Transaction was started.	

Tool validations
<p>* Step 5:</p> <p>Message TransactionEventRequest</p> <ul style="list-style-type: none"> - offline must be <i>true</i> - TriggerReason must be <i>EVDetected</i> <p>Post scenario validations:</p> <p>N/a</p>

TC_C_24_CS: Offline authorization through local authorization list - Expired

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)	<ul style="list-style-type: none"> - LocalAuthListCtrlr.LocalAuthListAvailable is implemented with value true - The Charging Station supports authorization methods other than NoAuthorization

Before (Preparations)
Configuration State: LocalAuthListEnabled is <i>true</i> (If implemented) LocalPreAuthorize is <i>true</i> (If implemented) OfflineTxForUnknownIdEnabled is <i>false</i> (If implemented) LocalAuthorizeOffline is <i>true</i> OfflineThreshold is <i><Configured RetryBackOffWaitMinimum_duration> + 60.0</i> RetryBackOffWaitMinimum is <i><Configured RetryBackOffWaitMinimum_duration></i> RetryBackOffRandomRange is <i>0</i>
Memory State: <i>IdTokenLocalAuthList</i> for <i><Configured expired idtoken fields></i>
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
1. The Test System closes the WebSocket connection AND does not accept a reconnect.	
2. <u>Manual Action</u> : Drive EV into parking bay.	
3. <u>Manual Action</u> : Present idToken.	
4. The Test System accepts reconnection attempt from the Charging Station.	
5. The Charging Stations sends a TransactionEventRequest	6. The Test System responds with a TransactionEventResponse
<u>Note(s)</u> : - The Charging Station will empty its Transaction message queue, this will only contain TransactionEventRequests if TxStartPoint was ParkingBayOccupancy and a Transaction was started.	

Tool validations
* Step 5: Message TransactionEventRequest - offline must be <i>true</i> - TriggerReason must be <i>EVDetected</i>
Post scenario validations: N/a

TC_C_25_CS: Offline authorization through local authorization list - Local Authorization List > Authorization Cache

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)	<ul style="list-style-type: none"> - LocalAuthListCtrlr.LocalAuthListAvailable is implemented with value true - AuthCacheCtrlr.Available 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 Test System closes the WebSocket connection AND does not accept a reconnect.	
<u>Manual Action</u> : Present idToken.	
<u>Note(s)</u> : The tool will wait for <i><Configured Transaction Duration></i> seconds.	
2. The Test System 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

TC_C_26_CS: Offline Authorization - Unknown Id

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)	<ul style="list-style-type: none"> - OfflineTxForUnknownIdEnabled is implemented. - The Charging Station supports authorization methods other than NoAuthorization

Before (Preparations)
Configuration State: LocalAuthListEnabled is <i>true</i> (If implemented) LocalPreAuthorize is <i>true</i> (If implemented) OfflineTxForUnknownIdEnabled is <i>true</i> LocalAuthorizeOffline is <i>true</i> MaxEnergyOnInvalidId is <i>0</i> (If implemented) StopTxOnInvalidId is <i>false</i>
Memory State: N/a
Reusable State(s): State is <i>StartOfflineTransaction</i>

Main (Test scenario)	
Charging Station	CSMS
1. The Charging Station notifies the CSMS about the current state of all connectors.	2. The Test System responds accordingly.
3. The Charging Stations sends a TransactionEventRequest	4. The Test System responds with a TransactionEventResponse
<u>Note(s):</u> - The Charging Station will empty its Transaction message queue. This will contain one or more TransactionEventRequest messages	<u>Note(s):</u> - The Test System will respond to the TransactionEventRequest containing the idToken, with idtokenInfo.status Invalid
5. Execute Reusable State <i>StopAuthorized</i>	
6. Execute Reusable State <i>EVConnectedPostSession</i>	
7. Execute Reusable State <i>EVDisconnected</i>	

Tool validations
<p>* Step 1:</p> <p>Message: StatusNotificationRequest</p> <ul style="list-style-type: none">- connectorStatus must be <i>Occupied</i> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none">- eventData[0].trigger must be <i>Delta</i>- eventData[0].actualValue must be <i>Occupied</i>- eventData[0].component.name must be <i>Connector</i>- eventData[0].variable.name must be <i>AvailabilityState</i> <p>* Step 3:</p> <p>All Message(s): TransactionEventRequest</p> <ul style="list-style-type: none">- offline must be <i>true</i> <p>* Step 4:</p> <p>One of the Message(s): TransactionEventRequest</p> <ul style="list-style-type: none">- chargingState must be <i>SuspendedEVSE</i>
<p>Post scenario validations:</p> <p>N/a</p>

TC_C_50_CS: Authorization using Contract Certificates 15118 - Online - Local contract certificate validation - Accepted

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:

TxCtrlr.TxStartPoint contains one or more of *PowerPathClosed*, *Authorized*, *EVConnected*, *ParkingBayOccupancy*

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*

Memory State:

CertificateInstalled for certificateType *V2GRootCertificate*

CertificateInstalled for certificateType *MORootCertificate*

RenewV2GChargingStationCertificate

Reusable State(s):

State is *EVConnectedPreSession*

Main (Test scenario)

Charging Station	CSMS
1. The Charging Station sends an <i>AuthorizeRequest</i> <u>Note(s):</u> <i>-The test case should be robust enough to also handle a <i>GetCertificateStatusRequest</i></i>	2. The Test System responds with a <i>AuthorizeResponse</i> with <i>idTokenInfo.status</i> <i>Accepted</i> and <i>certificateStatus</i> = <i>Accepted</i>
3. The Charging Station sends a <i>TransactionEventRequest</i>	4. The Test System responds with a <i>TransactionEventResponse</i> With <i>idTokenInfo.status</i> <i>Accepted</i>
5. Execute Reusable State <i>EnergyTransferStarted</i>	

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

TC_C_51_CS: Authorization using Contract Certificates 15118 - Online - Local contract certificate validation - Rejected

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)
<p>Configuration State: TxCtrlr.TxStartPoint contains one or more of <i>PowerPathClosed, Authorized, EVConnected, ParkingBayOccupancy</i> AuthCtrlr.AuthEnabled is <i>true</i> (If implemented AND ReadWrite) AuthCtrlr.DisableRemoteAuthorization is <i>false</i> (If implemented) For the ISO15118Ctrlr of the EVSE used in the PnC transaction: ISO15118Ctrlr.CentralContractValidationAllowed is <i>false</i> ISO15118Ctrlr.PnCEnabled is <i>true</i></p> <p>Memory State: <i>CertificateInstalled</i> for certificateType <i>V2GRootCertificate</i> <i>CertificateInstalled</i> for certificateType <i>MORootCertificate</i> <i>RenewV2GChargingStationCertificate</i></p> <p>Reusable State(s): State is <i>EVConnectedPreSession</i></p>

Main (Test scenario)	
Charging Station	CSMS
1. The Charging Station sends an AuthorizeRequest <u>Note(s):</u> <i>-The test case should be robust enough to also handle a GetCertificateStatusRequest</i>	2. The Test System responds with a AuthorizeResponse with idTokenInfo.status <i>Invalid</i> and certificateStatus = <i>ContractCancelled</i>

Tool validations
<p>* Step 1: Message: AuthorizeRequest - idToken.type must be <i>eMAID</i> - iso15118CertificateHashData is provided</p> <p>Post scenario validations: EV is not authorized and shall not charge: Charging Station does not send TransactionEventRequest with: - triggerReason = <i>Authorized</i> or chargingState = <i>Charging</i></p>

TC_C_52_CS: Authorization using Contract Certificates 15118 - Online - Central contract certificate validation - Accepted

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 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)
<p>Configuration State: TxCtrlr.TxStartPoint contains one or more of <i>PowerPathClosed</i>, <i>Authorized</i>, <i>EVConnected</i>, <i>ParkingBayOccupancy</i> AuthCtrlr.AuthEnabled is <i>true</i> (If implemented AND ReadWrite) AuthCtrlr.DisableRemoteAuthorization is <i>false</i> (If implemented) For the ISO15118Ctrlr of the EVSE used for the PnC transaction: ISO15118Ctrlr.CentralContractValidationAllowed is <i>true</i> ISO15118Ctrlr.ContractCertificateInstallationEnabled is <i>true</i> ISO15118Ctrlr.V2GCertificateInstallationEnabled is <i>true</i> ISO15118Ctrlr.PnCEnabled is <i>true</i></p> <p>Memory State: CertificateInstalled for certificateType <i>V2GRootCertificate</i> RenewV2GChargingStationCertificate</p> <p>Reusable State(s): State is EVConnectedPreSession</p>

Main (Test scenario)	
Charging Station	CSMS
1. The Charging Station sends an AuthorizeRequest <u>Note(s)</u> : -The test case should be robust enough to also handle a GetCertificateStatusRequest	2. The Test System responds with a AuthorizeResponse with idTokenInfo.status <i>Accepted</i> and certificateStatus = <i>Accepted</i>
3. The Charging Station sends a TransactionEventRequest	4. The Test System responds with a TransactionEventResponse With idTokenInfo.status <i>Accepted</i>
5. Execute Reusable State <i>EnergyTransferStarted</i>	

Tool validations
<p>* Step 1: Message: AuthorizeRequest - idToken.type must be <i>eMAID</i> - iso15118CertificateHashData may be provided - certificate is provided</p> <p>* Step 3: Message: TransactionEventRequest - triggerReason must be <i>Authorized</i></p> <p>Post scenario validations: N/a</p>

TC_C_53_CS: Authorization using Contract Certificates 15118 - Online - Central contract validation fails

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)	<ul style="list-style-type: none"> - The 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)
<p>Configuration State:</p> <p>TxCtrlr.TxStartPoint contains one or more of <i>PowerPathClosed</i>, <i>Authorized</i>, <i>EVConnected</i>, <i>ParkingBayOccupancy</i></p> <p>AuthCtrlr.AuthEnabled is <i>true</i> (If implemented AND ReadWrite)</p> <p>AuthCtrlr.DisableRemoteAuthorization is <i>false</i> (If implemented)</p> <p>For the ISO15118Ctrlr of the EVSE involved in the PnC transaction:</p> <p>ISO15118Ctrlr.CentralContractValidationAllowed is <i>true</i></p> <p>ISO15118Ctrlr.PnCEnabled is <i>true</i></p> <p>Memory State:</p> <p><i>CertificateInstalled</i> for certificateType <i>V2GRootCertificate</i></p> <p><i>RenewV2GChargingStationCertificate</i></p> <p>Reusable State(s):</p> <p>State is <i>EVConnectedPreSession</i></p>

Main (Test scenario)	
Charging Station	CSMS
<p>1. The Charging Station sends an AuthorizeRequest.</p> <p><u>Note(s):</u> -The test case should be robust enough to also handle a GetCertificateStatusRequest</p>	<p>2. The Test System responds with a AuthorizeResponse with idTokenInfo.status <i>Invalid</i> and certificateStatus = <i>CertificateRevoked</i></p>

Tool validations
<p>* Step 1:</p> <p>Message: AuthorizeRequest</p> <ul style="list-style-type: none"> - idToken.type must be <i>eMAID</i> - iso15118CertificateHashData may be provided - certificate is provided <p>Post scenario validations:</p> <p>EV is not authorized and shall not charge:</p> <p>Charging Station does not send TransactionEventRequest with:</p> <ul style="list-style-type: none"> - triggerReason = <i>Authorized</i> or chargingState = <i>Charging</i>

TC_C_54_CS: Authorization using Contract Certificates 15118 - Offline - ContractValidationOffline is true

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 for an EMAID that exists in authorization cache or local authorization list, while offline.
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:

TxCtrlr.TxStartPoint contains one or more of *PowerPathClosed*, *Authorized*, *EVConnected*, *ParkingBayOccupancy*

AuthCtrlr.AuthEnabled is *true* (If implemented AND ReadWrite)

AuthCacheCtrlr.Enabled is *true* OR **LocalAuthListCtrlr.Enabled** is *true*

OfflineTxForUnknownIdEnabled is *false* (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*

RenewV2GChargingStationCertificate

IdTokenCached15118 for *<Configured valid ISO 15118 IdToken>* (If implemented)

IdTokenLocalAuthList for *<Configured valid ISO 15118 IdToken>* (If implemented)

Reusable State(s):

N/a

Main (Test scenario)

Charging Station	CSMS
1. The Test System closes the WebSocket connection AND does not accept a reconnect.	
<u>Manual Action:</u> Drive EV into parking bay.	
<u>Manual Action:</u> Connect the EV and EVSE.	
<u>Notes(s):</u> The Test System will wait for <i><Configured Transaction Duration></i> seconds.	
2. The Test System accepts the reconnection attempt from the Charging Station after <i><Configured Transaction Duration></i> seconds.	
3. The Charging Station notifies the CSMS about the status change of the connector.	4. The Test System responds accordingly.
5. The Charging Station sends a TransactionEventRequest <u>Note(s):</u> - This step needs to be executed when TxStartPoint contains <i>EVConnected</i> OR the transaction already started (in the case TxStartPoint contains <i>ParkingBayOccupancy</i>)	6. The Test System responds with a TransactionEventResponse
7. The Charging Station sends a TransactionEventRequest	8. The Test System responds with a TransactionEventResponse With idTokenInfo.status <i>Accepted</i>

Main (Test scenario)
9. Execute Reusable State <i>EnergyTransferStarted</i>
<div><div>Tool validations</div><div><div>* Step 3:</div><div>Message: StatusNotificationRequest</div><div>- connectorStatus must be <i>Occupied</i></div><div>Message: NotifyEventRequest</div><div>- eventData[0].trigger must be <i>Delta</i></div><div>- eventData[0].actualValue must be <i>Occupied</i></div><div>- eventData[0].component.name must be <i>Connector</i></div><div>- eventData[0].variable.name must be <i>AvailabilityState</i></div><div>* Step 5:</div><div>Message: TransactionEventRequest</div><div>- triggerReason must be <i>CablePluggedIn</i></div><div>- transactionInfo.chargingState must be <i>EVConnected</i></div><div>- offline <i>true</i></div><div>* Step 7:</div><div>Message: TransactionEventRequest</div><div>- triggerReason must be <i>Authorized</i></div><div>- offline <i>true</i></div></div><div><div>Post scenario validations:</div><div>N/a</div></div></div>

TC_C_55_CS: Authorization using Contract Certificates 15118 - Offline - ContractValidationOffline is false

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 will not authorize with contract certificates when offline.
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

<p>Before (Preparations)</p> <p>Configuration State: TxCtrlr.TxStartPoint contains one or more of <i>PowerPathClosed</i>, <i>Authorized</i>, <i>EVConnected</i>, <i>ParkingBayOccupancy</i> AuthCtrlr.AuthEnabled is <i>true</i> (If implemented AND ReadWrite) AuthCacheCtrlr.Enabled is <i>true</i> OR LocalAuthListCtrlr.Enabled is <i>true</i> OfflineTxForUnknownIdEnabled is <i>false</i> (If implemented) OfflineThreshold is <i><Configured RetryBackOffWaitMinimum_duration> + 60.0</i> RetryBackOffWaitMinimum is <i><Configured RetryBackOffWaitMinimum_duration></i> RetryBackOffRandomRange is <i>0</i> <u>Note:</u> <i><Configured RetryBackOffWaitMinimum_duration> should be long enough to execute manual tasks></i> For the ISO15118Ctrlr of the EVSE involved in the PnC transaction: ISO15118Ctrlr.ContractValidationOffline is <i>false</i> ISO15118Ctrlr.PnCEnabled is <i>true</i></p> <p>Memory State: <i>CertificateInstalled</i> for certificateType <i>V2GRootCertificate</i> <i>CertificateInstalled</i> for certificateType <i>MORootCertificate</i> <i>RenewV2GChargingStationCertificate</i> <i>IdTokenCached15118</i> for <i><Configured valid ISO 15118 IdToken></i> (If implemented) <i>IdTokenLocalAuthList</i> for <i><Configured valid ISO 15118 IdToken></i> (If implemented)</p> <p>Reusable State(s): N/a</p>

Main (Test scenario)	
Charging Station	CSMS
1. The Test System closes the WebSocket connection AND does not accept a reconnect.	
<u>Manual Action:</u> Drive EV into parking bay.	
<u>Manual Action:</u> Connect the EV and EVSE.	
<u>Note(s):</u> The tool will wait for <i><Configured Transaction Duration></i> seconds.	
2. The Test System accepts the reconnection attempt from the Charging Station after <i><Configured Transaction Duration></i> seconds.	
3. The Charging Station notifies the CSMS about the status change of the connector.	4. The Test System responds accordingly.
5. The Charging Station sends a TransactionEventRequest <u>Note(s):</u> - This step needs to be executed when TxStartPoint contains <i>EVConnected</i> OR the transaction already started (in the case TxStartPoint contains <i>ParkingBayOccupancy</i>)	6. The Test System responds with a TransactionEventResponse
7. The Charging Station has NOT started charging and does NOT send TransactionEventRequest message(s) with triggerReason <i>Authorized</i> OR <i>ChargingStateChanged</i> .	

Tool validations
<p>* Step 3:</p> <p>Message: StatusNotificationRequest</p> <ul style="list-style-type: none">- connectorStatus must be <i>Occupied</i> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none">- eventData[0].trigger must be <i>Delta</i>- eventData[0].actualValue must be <i>Occupied</i>- eventData[0].component.name must be <i>Connector</i>- eventData[0].variable.name must be <i>AvailabilityState</i> <p>* Step 5:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none">- triggerReason must be <i>CablePluggedIn</i>- transactionInfo.chargingState must be <i>EVConnected</i>- offline <i>true</i>
<p>Post scenario validations:</p> <p>EV is not authorized and shall not charge:</p> <p>Charging Station does not send TransactionEventRequest with:</p> <ul style="list-style-type: none">- triggerReason = <i>Authorized</i> or chargingState = <i>Charging</i>

TC_C_56_CS: Local start transaction - Authorization Unknown

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 <i>true</i> (If implemented AND ReadWrite) AuthCtrlr.DisableRemoteAuthorization is <i>false</i> (If implemented)
Memory State: N/a
Reusable State(s): State is <i>EVConnectedPreSession</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action</u> : Present invalid idToken.	
1. The Charging Station sends an AuthorizeRequest	2. The Test System responds with an AuthorizeResponse with idTokenInfo.status <i>Unknown</i>
<u>Note(s)</u> : - The Charging Station SHALL NOT send a TransactionEventRequest message after the AuthorizeRequest from step 1. - The Test System 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

TC_C_100_CS: Local start transaction - Authorization first - Cable plugin timeout

Test case name	Local start transaction - Authorization first - Cable plugin timeout
Test case Id	TC_C_100_CS
Use case Id(s)	C01
Requirement(s)	C01.FR.26
System under test	Charging Station
Description	OCPP 2.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 <i>Authorized</i> (local)

Main (Test scenario)	
Charging Station	CSMS
<p>1. The Charging Station sends a TransactionEventRequest</p> <p><u>Note(s):</u> - This step only needs to be executed after the <Configured ev_connection_timeout> expires, if the transaction has been started. So in the case TxStartPoint contains ParkingBayOccupancy OR Authorized</p>	<p>2. The Test System responds with a TransactionEventResponse</p>
<p><u>Note(s):</u> - Optionally the Charging Station can send a StatusNotificationRequest or NotifyEventRequest with status Available</p>	
<p><u>Manual Action:</u> Connect the EV and EVSE.</p>	
<p><u>Note(s):</u> - This step only needs to be executed after the <Configured EV connection timeout> expires, if the transaction has been started. So in the case TxStartPoint contains ParkingBayOccupancy OR Authorized</p>	

Tool validations
* Step 1: Message: TransactionEventRequest - triggerReason must be <i>EVConnectTimeout</i> If <Configured TxStopPoint> contains <i>Authorized</i> then eventType must be <i>Ended</i> AND transactionInfo.stoppedReason must be <i>Timeout</i> Else eventType must be <i>Updated</i>
Post scenario validations: - Power transfer will not be started

TC_C_103_CS: Authorization with prepaid card - success

Test case name	Authorization with prepaid card - success
Test case Id	TC_C_103_CS
Use case Id(s)	C17
Requirement(s)	C17.FR.04
System under test	CS
Description	This test case verifies if the CS communicates the transaction limits correctly.
Purpose	
Prerequisite(s)	<ul style="list-style-type: none"> - Prepaid card is available. - TxCtrlr.SupportedLimits contains <i>MaxCost</i>

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> : EV driver presents prepaid card	
1. The CS sends a AuthorizeRequest	2. The Test System responds with an AuthorizeResponse with - idTokenInfo.status <i>Accepted</i> - idTokenInfo.cacheExpiryDateTime <i><current date/time></i>
<u>Note</u> : Only when TxStartPoint does not contain ParkingBayOccupancy or Authorized: - <u>Manual action</u> : Plug cable in	
3. The CS sends a TransactionEventRequest	4. The Test System responds with a TransactionEventResponse with - idTokenInfo.status <i>Accepted</i> - idTokenInfo.cacheExpiryDateTime <i><current date/time></i> - transactionLimit.maxCost = 123.32

Tool validations
* Step 1: Message AuthorizeRequest idToken.idToken <i><prepaid card id></i> idToken.type = "ISO14443" or "ISO15693"
* Step 3: Message TransactionEventRequest - eventType <i>Updated</i> (for TxStartPoint <i>ParkingBayOccupancy</i>) or <i>Started</i> (for all other TxStartPoints) - idToken.idToken <i><prepaid card id></i> - idToken.type = <i><idTokenType from step 1></i>
Post scenario validations: - transactionInfo.chargingState must transition to <i>Charging</i>

TC_C_104_CS: Authorization with prepaid card - no credit

Test case name	Authorization with prepaid card - no credit
Test case Id	TC_C_104_CS
Use case Id(s)	C17
Requirement(s)	C17.FR.05
System under test	CS
Description	This test case verifies if the CS communicates the transaction limits correctly.
Purpose	To verify that the Charging Station is able to handle when a prepaid card has no credit.
Prerequisite(s)	<ul style="list-style-type: none"> - Prepaid card is available. - TxCtrlr.SupportedLimits contains <i>MaxCost</i>

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>EVConnectedPreSession</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action:</u> <i>EV driver presents prepaid card</i>	
1. The CS sends a AuthorizeRequest	2. The Test System responds with an AuthorizeResponse with - idTokenInfo.status <i>NoCredit</i> - idTokenInfo.cacheExpiryDateTime <i><current date/time></i>

Tool validations
* Step 1: Message AuthorizeRequest idToken.idToken <i><prepaid card id></i> idToken.type = "ISO14443" or "ISO15693"
Post scenario validations: <ul style="list-style-type: none"> - The Charging Station must not send a TransactionEventRequest, unless TxStartPoint contains <i>ParkingBayOccupancy</i>. - The Charging Station must not offer energy to the EV.

TC_C_105_CS: Integrated Payment Terminal - CSMS rejects authorization

Test case name	Integrated Payment Terminal - CSMS rejects authorization
Test case Id	TC_C_105_CS
Use case Id(s)	C18
Requirement(s)	C18.FR.01, C18.FR.02, C18.FR.04, C18.FR.07
System under test	Charging Station
Description	To start/authorize a transaction from a payment terminal connected directly to the Charging Station.
Purpose	To verify that the Charging Station correctly handles payment terminal authorizations when the CSMS rejects the authorization.
Prerequisite(s)	- Charging Station has an integrated Payment Terminal

Before (Preparations)
Configuration State: - PaymentCtrlr.Enabled = true - PaymentCtrlr.AuthorizationAmount = 50.00 - PaymentCtrlr.AuthorizeDirectPayment = true - PaymentCtrlr.PaymentDetails = CardLast4Digits
Memory State: N/a
Reusable State(s): State is <i>DefaultTariffConfigure</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action</u> : Present a payment card to the payment terminal	
<u>Note(s)</u> : - The Charging Station instructs payment terminal to request authorization for this card from PSP for amount of 50.00 EUR - PSP authorizes the payment card and provides <PspRef>	
1. The Charging Station sends an AuthorizeRequest	2. The Test System responds with an AuthorizeResponse with idTokenInfo.status = <i>Invalid</i>
<u>Note(s)</u> : - <i>The Charging Station instructs payment terminal to release the authorization amount and cancel the payment</i>	

Tool validations
* Step 1: Message AuthorizeRequest - idToken must be <Not empty> - idToken.type must be <i>DirectPayment</i> - idToken.additionalInfo[0].additionalIdToken must be <not omitted> - idToken.additionalInfo[0].type must be <i>CardLast4Digits</i>
Post scenario validations: - Authorization amount is released and the payment is cancelled.

TC_C_106_CS: Integrated Payment Terminal - Only Payment Terminal authorises

Test case name	Integrated Payment Terminal - Only Payment Terminal authorises
Test case Id	TC_C_106_CS
Use case Id(s)	C18
Requirement(s)	C18.FR.01, C18.FR.02, C18.FR.03, C18.FR.06
System under test	Charging Station
Description	To start/authorize a transaction from a payment terminal connected directly to the Charging Station.
Purpose	To verify that the Charging Station can authorize a transaction using only a payment terminal
Prerequisite(s)	- Charging Station has an integrated Payment Terminal

Before (Preparations)
Configuration State: - PaymentCtrlr.Enabled = <i>true</i> - *PaymentCtrlr.AuthorizationAmount = <i>25.00</i> - PaymentCtrlr.AuthorizeDirectPayment = <i>false</i> - PaymentCtrlr.PaymentDetails = <i>CardLast4Digits</i>
Memory State: N/a
Reusable State(s): And <i>DefaultTariffConfigured</i>

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>AuthorizedPaymentTerminal</i> (Local)	
2. Execute Reusable State <i>EVConnectedPreSession</i>	
3. Execute Reusable State <i>EnergyTransferStarted</i>	

Tool validations
* Step 1 and 2 and 3 combined: In the first TransactionEventRequest message that contains idToken - idToken must be <i><Not empty></i> - idToken.type must be <i>DirectPayment</i> - idToken.additionalInfo[0].additionalIdToken must be <i><not omitted></i> - idToken.additionalInfo[0].type must be <i>CardLast4Digits</i> - transactionInfo.transactionLimit.maxCost must be <i>25.00 EUR</i>
Post scenario validations: N/a

TC_C_107_CS: Integrated Payment Terminal - Payment Terminal and CSMS authorises

Test case name	Integrated Payment Terminal - Payment Terminal and CSMS authorises
Test case Id	TC_C_107_CS
Use case Id(s)	C18
Requirement(s)	C18.FR.01, C18.FR.02, C18.FR.04, C18.FR.05, C18.FR.06
System under test	Charging Station
Description	To start/authorize a transaction from a payment terminal connected directly to the Charging Station.
Purpose	To verify if the Charging Station is able to handle authorization using a locally connected payment terminal.
Prerequisite(s)	- Charging Station has an integrated Payment Terminal

Before (Preparations)
Configuration State: - PaymentCtrlr.Enabled = <i>true</i> - *PaymentCtrlr.AuthorizationAmount = 25.00 - PaymentCtrlr.AuthorizeDirectPayment = <i>true</i> - PaymentCtrlr.PaymentDetails = <i>CardLast4Digits</i>
Memory State: N/a
Reusable State(s): State is <i>DefaultTariffConfigured</i>

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>AuthorizedPaymentTerminal</i> (Local)	
2. Execute Reusable State <i>EVConnectedPreSession</i>	
3. Execute Reusable State <i>EnergyTransferStarted</i>	

Tool validations
* Step 1, 2 and 3 combined: In the first TransactionEventRequest message that contains idToken - idToken must be <i><Not empty></i> - idToken.type must be <i>DirectPayment</i> - idToken.additionalInfo[0].additionalIdToken must be <i><not omitted></i> - idToken.additionalInfo[0].type must be <i>CardLast4Digits</i> - transactionInfo.transactionLimit.maxCost must be 25.00
Post scenario validations: - N/a

TC_C_108_CS: Integrated Payment Terminal - VAT number validation

Test case name	Integrated Payment Terminal - VAT number validation
Test case Id	TC_C_108_CS
Use case Id(s)	C18
Requirement(s)	C18.FR.08
System under test	Charging Station
Description	To start/authorize a transaction from a payment terminal connected directly to the Charging Station.
Purpose	To verify that the Charging Station can let CSMS validate VAT numbers.
Prerequisite(s)	- Charging Station supports VAT number validation - Charging Station has an integrated Payment Terminal

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> : EV Driver types in a VAT number	
1. The Charging Station sends a VatNumberValidationRequest	2. The Test System responds with a VatNumberValidationResponse with vatNumber = <invalidVatNumber of step 1> evseld = <VatNumberValidationRequest.evseld of step 1> status = <i>Rejected</i>
<u>Manual Action</u> : EV Driver types in a VAT number	
3. The Charging Station sends a VatNumberValidationRequest	4. The Test System responds with a VatNumberValidationResponse with vatNumber = <validVatNumber of step 3> evseld = <VatNumberValidationRequest.evseld of step 3> status = <i>Accepted</i>

Tool validations
* Step 1: Message VatNumberValidationRequest - evseld must be <evse id> or <omitted>
* Step 3: Message VatNumberValidationRequest - evseld must be <evse id> or <omitted>
Post scenario validations: N/a

TC_C_109_CS: Integrated Payment Terminal - Cancellation prior to transaction - Only Payment Terminal authorised - EVConnectionTimeout

Test case name	Integrated Payment Terminal - Cancellation prior to transaction - Only Payment Terminal authorised - EVConnectionTimeout
Test case Id	TC_C_109_CS
Use case Id(s)	C19
Requirement(s)	C18.FR.01, C18.FR.02, C18.FR.03, C19.FR.01
System under test	Charging Station
Description	In order to test that Charging Station supports cancellation of a payment terminal authorization when no EV is connected within the timeout period.
Purpose	To verify if the Charging Station is able to handle cancellation of a payment terminal authorization when no EV is connected within the timeout period.
Prerequisite(s)	- Charging Station has an integrated Payment Terminal - TxStartPoint does NOT contain <i>ParkingBayOccupancy</i> or <i>Authorized</i>

Before (Preparations)
Configuration State: PaymentCtrlr.Enabled is <i>true</i> PaymentCtrlr.AuthorizationAmount is <i>25.00</i> PaymentCtrlr.AuthorizeDirectPayment is <i>false</i> TxCtrlr.EVConnectionTimeout is <i>5</i> TxStartPoint <i>EVConnected</i> or <i>PowerPathClosed</i> or <i>EnergyTransfer</i> or <i>DataSigned</i>
Memory State: N/a
Reusable State(s): State is <i>DefaultTariffConfigured</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action</u> : Present a payment card to the payment terminal.	
<u>Note(s)</u> : <ul style="list-style-type: none">- Charging Station instructs payment terminal to request authorization for this card from PSP for amount of 25.00 EUR- PSP authorizes the payment card and provides <PspRef>- Charging Station authorizes EV to charge- The EVConnectionTimeout occurs after 5 seconds- The Charging Station instructs payment terminal to release the authorization amount of 25.00 EUR and cancel the payment	

Tool validations
N/a
Post scenario validations: - The authorization amount of 25.00 EUR is released and the payment is cancelled. - The Charging Station did NOT send a AuthorizeRequest, TransactionEventRequest(eventType=Started) and NotifySettlementRequest.

TC_C_110_CS: Integrated Payment Terminal - Cancellation prior to transaction - Payment Terminal and CSMS authorised - stopped by EV driver

Test case name	Integrated Payment Terminal - Cancellation prior to transaction - Payment Terminal and CSMS authorised - stopped by EV driver
Test case Id	TC_C_110_CS
Use case Id(s)	C19
Requirement(s)	C18.FR.01, C18.FR.02, C18.FR.05, C19.FR.01, C19.FR.02
System under test	Charging Station
Description	In order to test that Charging Station supports cancellation of a payment terminal authorization when the EV driver stops the session at the Charging Station.
Purpose	To verify if the Charging Station is able to handle cancellation of a payment terminal authorization when the EV driver stops the session at the Charging Station.
Prerequisite(s)	- Charging Station has an integrated Payment Terminal - TxStartPoint does NOT contain <i>ParkingBayOccupancy</i> or <i>Authorized</i>

Before (Preparations)
Configuration State: PaymentCtrlr.Enabled is <i>true</i> PaymentCtrlr.AuthorizationAmount is 25.00 PaymentCtrlr.AuthorizeDirectPayment is <i>true</i> TxStartPoint <i>EVConnected</i> or <i>PowerPathClosed</i> or <i>EnergyTransfer</i> or <i>DataSigned</i>
Memory State: N/a
Reusable State(s): State is <i>DefaultTariffConfigured</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action:</u> Present a payment card to the payment terminal.	
<u>Note(s):</u> - Charging Station instructs payment terminal to request authorization for this card from PSP for amount of 25.00 EUR - PSP authorizes the payment card and provides <PspRef> - Charging Station authorizes EV to charge	
1. The Charging Station sends an AuthorizeRequest	2. The Test System responds with an AuthorizeResponse with status <i>Accepted</i>
<u>Manual Action:</u> EV Driver stops the session at Charging Station.	
<u>Note(s):</u> - Charging Station instructs payment terminal to release the authorization amount of 25.00 EUR and cancel the payment	
3. The Charging Station sends a NotifySettlementRequest	4. The Test System responds with a NotifySettlementResponse

Tool validations
* Step 3: Message NotifySettlementRequest - transactionId must be <omitted> - pspRef must be <Equal to idToken from AuthorizeRequest> - status must be <i>Canceled</i> - settlementTime must be <near current time>
Post scenario validations: - The authorization amount of 25.00 EUR is released and the payment is cancelled.

TC_C_111_CS: Integrated Payment Terminal - Cancellation prior to transaction - Only Payment Terminal authorised - stopped by EV driver

Test case name	Integrated Payment Terminal - Cancellation prior to transaction - Only Payment Terminal authorised - stopped by EV driver
Test case Id	TC_C_111_CS
Use case Id(s)	C19
Requirement(s)	C18.FR.01, C18.FR.02, C18.FR.03, C19.FR.01
System under test	Charging Station
Description	In order to test that Charging Station supports cancellation of a payment terminal authorization when the EV driver stops the session at the Charging Station.
Purpose	To verify if the Charging Station is able to handle cancellation of a payment terminal authorization when the EV driver stops the session at the Charging Station.
Prerequisite(s)	- Charging Station has an integrated Payment Terminal - TxStartPoint does NOT contain <i>ParkingBayOccupancy</i> or <i>Authorized</i>

Before (Preparations)
Configuration State: PaymentCtrlr.Enabled is <i>true</i> PaymentCtrlr.AuthorizationAmount is 25.00 PaymentCtrlr.AuthorizeDirectPayment is <i>false</i> TxStartPoint <i>EVConnected</i> or <i>PowerPathClosed</i> or <i>EnergyTransfer</i> or <i>DataSigned</i>
Memory State: N/a
Reusable State(s): State is <i>DefaultTariffConfigured</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action</u> : Present a payment card to the payment terminal.	
<u>Note(s)</u> : - Charging Station instructs payment terminal to request authorization for this card from PSP for amount of 25.00 EUR - PSP authorizes the payment card and provides <PspRef> - Charging Station authorizes EV to charge	
<u>Manual Action</u> : EV Driver stops the session at Charging Station.	
<u>Note(s)</u> : - Charging Station instructs payment terminal to release the authorization amount of 25.00 EUR and cancel the payment - Charging Station does not send any NotifySettlementRequest	

Tool validations
N/a
Post scenario validations: - The authorization amount of 25.00 EUR is released and the payment is cancelled. - The Charging Station does not send any <i>NotifySettlementRequest</i> .

TC_C_112_CS: Integrated Payment Terminal- Cancellation prior to transaction - Payment Terminal and CSMS authorised - EVConnectionTimeout

Test case name	Integrated Payment Terminal - Cancellation prior to transaction - Payment Terminal and CSMS authorised - EVConnectionTimeout
Test case Id	TC_C_112_CS
Use case Id(s)	C19
Requirement(s)	C18.FR.01, C18.FR.02, C18.FR.05, C19.FR.01, C19.FR.02
System under test	Charging Station
Description	In order to test that Charging Station supports cancellation of a payment terminal authorization when no EV is connected within the timeout period.
Purpose	To verify if the Charging Station is able to handle cancellation of a payment terminal authorization when no EV is connected within the timeout period.
Prerequisite(s)	- Charging Station has an integrated Payment Terminal - TxStartPoint does NOT contain <i>ParkingBayOccupancy</i> or <i>Authorized</i>

Before (Preparations)
Configuration State: PaymentCtrlr.Enabled is <i>true</i> PaymentCtrlr.AuthorizationAmount is <i>50.00</i> PaymentCtrlr.AuthorizeDirectPayment is <i>true</i> TxCtrlr.EVConnectionTimeout is <i>5</i> TxStartPoint <i>EVConnected</i> or <i>PowerPathClosed</i> or <i>EnergyTransfer</i> or <i>DataSigned</i>
Memory State: N/a
Reusable State(s): State is <i>DefaultTariffConfigured</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action:</u> Present a payment card to the payment terminal.	
<u>Note(s):</u> - Charging Station instructs payment terminal to request authorization for this card from PSP for amount of 50.00 - PSP authorizes the payment card and provides <PspRef>	
1. The Charging Station sends an AuthorizeRequest	2. The Test System responds with an AuthorizeResponse with status <i>Accepted</i>
<u>Note(s):</u> - Charging Station authorizes EV to charge - The EVConnectionTimeout occurs after 5 seconds - Charging Station instructs payment terminal to release the authorization amount of 25.00 EUR and cancel the payment	
3. The Charging Station sends a NotifySettlementRequest	4. The Test System responds with a NotifySettlementResponse

Tool validations

* Step 1:

Message **AuthorizeRequest**

- **idToken** must be *<Not empty>*
- **idToken.type** must be *DirectPayment*

* Step 3:

Message **NotifySettlementRequest**

- **transactionId** must be *<omitted>*
- **pspRef** must be *<Equal to idToken from AuthorizeRequest>*
- **status** must be *Canceled*
- **settlementTime** must be *<near current time>*

Post scenario validations:

- The authorization amount of 50.00 EUR is released and the payment is cancelled.

TC_C_113_CS: Integrated Payment Terminal - Cancellation after start of transaction - stopped by EV driver

Test case name	Integrated Payment Terminal - Cancellation after start of transaction - stopped by EV driver
Test case Id	TC_C_113_CS
Use case Id(s)	C20
Requirement(s)	C20.FR.01, C20.FR.03, C20.FR.04
System under test	Charging Station
Description	In order to test that Charging Station supports cancellation of a payment terminal authorization after a transaction has started when the EV driver stops the session at the Charging Station.
Purpose	To verify if the Charging Station is able to handle cancellation of a payment terminal authorization after a transaction has started when the EV driver stops the session at the Charging Station.
Prerequisite(s)	- Charging Station has an integrated Payment Terminal - TxStartPoint contains <i>ParkingBayOccupancy</i> or <i>Authorized</i>

Before (Preparations)
Configuration State: PaymentCtrlr.Enabled is <i>true</i> PaymentCtrlr.AuthorizationAmount is <i>50.00</i> PaymentCtrlr.AuthorizeDirectPayment is <i>false</i> PaymentCtrlr.PaymentDetails = <i>CardLast4Digits</i> PaymentCtrlr.SettlementByCSMS is <i>false</i> TxCtrlr.EVConnectionTimeout is <i>5</i> TxStartPoint <i>Authorized</i> or <i>ParkingBayOccupancy</i>
Memory State: N/a
Reusable State(s): State is <i>DefaultTariffConfigured</i>

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>AuthorizedPaymentTerminal</i> (Local)	
<u>Note(s)</u> : - The <i>EVConnectionTimeout</i> occurs after 5 seconds - Charging Station instructs payment terminal to release the authorization amount of 25.00 EUR and cancel the payment	
2. The Charging Station sends a TransactionEventRequest	3. The Test System responds with a TransactionEventResponse
4. The Charging Station sends a NotifySettlementRequest	5. The Test System responds with a NotifySettlementResponse

Tool validations

* Step 1: (First *TransactionEventRequest* with *idToken*)

Message **TransactionEventRequest**

- **transactionInfo.transactionLimit.maxCost** must be *50.00*

* Step 2:

A message **TransactionEventRequest**

- **eventType** is *Ended*

- **triggerReason** is *EVConnectionTimeOut*

* Step 4:

Message **NotifySettlementRequest**

- **transactionId** must be *<transactionId>*

- **pspRef** must be *<Equal to idToken from TransactionEventRequest with triggerReason Authorized or RemoteStart>*

- **status** must be *Canceled*

- **settlementTime** must be *<near current time>*

Post scenario validations:

- The authorization amount of 50.00 EUR is released and the payment is cancelled.

TC_C_114_CS: Settlement at end of transaction - settled by CS, receipt by CS

Test case name	Settlement at end of transaction - settled by CS, receipt by CS
Test case Id	TC_C_114_CS
Use case Id(s)	C21
Requirement(s)	C21.FR.01, C21.FR.02, C21.FR.04
System under test	Charging Station
Description	In order to test that Charging Station supports settlement at the end of a transaction where the settlement is handled by the Charging Station and the receipt is provided by the Charging Station.
Purpose	To verify if the Charging Station is able to handle settlement at the end of a transaction where the settlement is handled by the Charging Station and the receipt is provided by the Charging Station.
Prerequisite(s)	- Charging Station has an integrated Payment Terminal

Before (Preparations)

Configuration State:

TariffCostCtrlr.Enabled[Tariff] is *true*

TariffCostCtrlr.Enabled[Cost] is *true*

TariffCostCtrlr.Currency is *EUR*

PaymentCtrlr.Enabled is *true*

PaymentCtrlr.AuthorizationAmount is *50.00*

PaymentCtrlr.AuthorizeDirectPayment is *false*

PaymentCtrlr.SettlementByCSMS is *false*

PaymentCtrlr.ReceiptByCSMS is *false*

Memory State:

N/a

Reusable State(s):

N/a

Main (Test scenario)

Charging Station	CSMS
2. The Charging Station responds with a SetDefaultTariffResponse	1. The Test System sends a SetDefaultTariffRequest with evseld 0 tariff.tariffId <i>Test System1</i> tariff.currency <i>EUR</i> tariff.fixedFee.taxRates[0].type <i>MyTax</i> tariff.fixedFee.taxRates[0].tax 21 tariff.fixedFee.prices.priceFixed[0] 15.00
3. Execute Reusable State <i>AuthorizedPaymentTerminal</i> (Local)	
4. Execute Reusable State <i>EVConnectedPreSession</i>	
5. Execute Reusable State <i>EnergyTransferStarted</i>	
6. Execute Reusable State <i>StopAuthorized</i> (Remote)	
7. Execute Reusable State <i>ParkingBayUnoccupied</i>	
Note(s): - Charging Station settles the total cost of the transaction via the payment terminal of: 18.15 EUR - Charging Station receives the settlement status from payment terminal - The NotifySettlementRequest is sent to the CSMS around the time the transaction ends, so depending on the TxStopPoint it can be transmitted during or after steps 6 and 7	
8. The Charging Station sends a NotifySettlementRequest	9. The Test System responds with a NotifySettlementResponse

Tool validations

* Step 2:

Message **SetDefaultTariffResponse**

- **status** must be *Accepted*

* Step 3 and 4 and 5 combined: (First transactionEventRequest with idToken)

Message **TransactionEventRequest** with:

- **idToken** must be *<Not empty>*
- **idToken.type** must be *DirectPayment*
- **transactionInfo.transactionId** must be *<transactionId>*
- **transactionInfo.transactionLimit.maxCost** must be *50.00*

* Step 6 and 7 combined (TransactionEventRequest with eventType Ended):

Message **TransactionEventRequest** with:

- **costDetails.totalCost.currency** must be *EUR*
- **costDetails.totalCost.fixed.exclTax** must be *15.00*
- **costDetails.totalCost.fixed.inclTax** must be *18.15 EUR*
- **costDetails.totalCost.fixed.taxRate.type** must be *MyTax*
- **costDetails.totalCost.fixed.taxRate.tax** must be *21*

* Step 8:

Message **NotifySettlementRequest**

- **transactionId** must be *<transactionId>*
- **pspRef** must be *<Equal to idToken from TransactionEventRequest with triggerReason Authorized or RemoteStart>*
- **status** must be *<not omitted>*
- **settlementTime** must be *<not omitted>*
- **settlementAmount** must be *18.15*
- **receiptUrl** must be *<not omitted>*

Post scenario validations:

- The transaction is settled successfully with the payment terminal for 18.15 EUR.

TC_C_115_CS: Settlement at end of transaction - settled by CSMS

Test case name	Settlement at end of transaction - settled by CSMS
Test case Id	TC_C_115_CS
Use case Id(s)	C21
Requirement(s)	C21.FR.06,
System under test	Charging Station
Description	In order to test that Charging Station supports settlement at the end of a transaction where the settlement is handled by the CSMS.
Purpose	To verify if the Charging Station is able to handle settlement at the end of a transaction where the settlement is handled by the CSMS.
Prerequisite(s)	- Charging Station has an integrated Payment Terminal

Before (Preparations)

Configuration State:

TariffCostCtrlr.Enabled[Tariff] is *true*

TariffCostCtrlr.Enabled[Cost] is *true*

TariffCostCtrlr.Currency is *EUR*

PaymentCtrlr.Enabled is *true*

PaymentCtrlr.AuthorizationAmount is *50.00*

PaymentCtrlr.AuthorizeDirectPayment is *false*

PaymentCtrlr.SettlementByCSMS is *true*

Memory State:

N/a

Reusable State(s):

N/a

Main (Test scenario)

Charging Station	CSMS
2. The Charging Station responds with a SetDefaultTariffResponse	1. The Test System sends a <code>SetDefaultTariffRequest</code> with evseld 0 tariff.tariffId Test System1 tariff.currency EUR tariff.fixedFee.taxRates[0].type MyTax tariff.fixedFee.taxRates[0].tax 21 tariff.fixedFee.prices.priceFixed[0] 15.00
3. Execute Reusable State <i>AuthorizedPaymentTerminal</i> (Local)	
4. Execute Reusable State <i>EVConnectedPreSession</i>	
5. Execute Reusable State <i>EnergyTransferStarted</i>	
6. Execute Reusable State <i>StopAuthorized</i> (Remote)	
7. Execute Reusable State <i>ParkingBayUnoccupied</i>	
Note(s): - Charging Station does not send any <code>NotifySettlementRequest</code> as settlement is handled by CSMS	

Tool validations

* Step 2:

Message **SetDefaultTariffResponse**

- **status** must be *Accepted*

* Step 3 and 4 and 5 combined: (First transactionEventRequest with idToken)

Message **TransactionEventRequest** with:

- **idToken** must be *<Not empty>*
- **idToken.type** must be *DirectPayment*
- **transactionInfo.transactionId** must be *<transactionId>*
- **transactionInfo.transactionLimit.maxCost** must be *50.00*

* Step 6 and 7 combined (TransactionEventRequest with eventType Ended):

Message **TransactionEventRequest** with:

- **costDetails.totalCost.currency** must be *EUR*
- **costDetails.totalCost.fixed.exclTax** must be *15.00*
- **costDetails.totalCost.fixed.inclTax** must be *18.15 EUR*
- **costDetails.totalCost.fixed.taxRate.type** must be *MyTax*
- **costDetails.totalCost.fixed.taxRate.tax** must be *21*

Post scenario validations:

- The Charging Station does not send any NotifySettlementRequest as the settlement is handled by the CSMS.

TC_C_116_CS: Settlement at end of transaction - settled by CS, receipt by CSMS

Test case name	Settlement at end of transaction - settled by CS, receipt by CSMS
Test case Id	TC_C_116_CS
Use case Id(s)	C21
Requirement(s)	C21.FR.01, C21.FR.02, C21.FR.04
System under test	Charging Station
Description	In order to test that Charging Station supports settlement at the end of a transaction where the settlement is handled by the Charging Station and the receipt is provided by the CSMS.
Purpose	To verify if the Charging Station is able to handle settlement at the end of a transaction where the settlement is handled by the Charging Station and the receipt is provided by the CSMS.
Prerequisite(s)	- Charging Station has an integrated Payment Terminal

Before (Preparations)

Configuration State:

TariffCostCtrlr.Enabled[Tariff] is *true*

TariffCostCtrlr.Enabled[Cost] is *true*

TariffCostCtrlr.Currency is *EUR*

PaymentCtrlr.Enabled is *true*

PaymentCtrlr.AuthorizationAmount is *50.00*

PaymentCtrlr.AuthorizeDirectPayment is *false*

PaymentCtrlr.SettlementByCSMS is *false*

PaymentCtrlr.ReceiptByCSMS is *true*

Memory State:

N/a

Reusable State(s):

N/a

Main (Test scenario)

Charging Station	CSMS
2. The Charging Station responds with a SetDefaultTariffResponse	1. The Test System sends a SetDefaultTariffRequest with evseld 0 tariff.tariffId <i>Test System1</i> tariff.currency <i>EUR</i> tariff.fixedFee.taxRates[0].type <i>MyTax</i> tariff.fixedFee.taxRates[0].tax 21 tariff.fixedFee.prices.priceFixed[0] 15.00
3. Execute Reusable State <i>AuthorizedPaymentTerminal</i> (Local)	
4. Execute Reusable State <i>EVConnectedPreSession</i>	
5. Execute Reusable State <i>EnergyTransferStarted</i>	
6. Execute Reusable State <i>StopAuthorized</i> (Remote)	
7. Execute Reusable State <i>ParkingBayUnoccupied</i>	
Note(s): - Charging Station settles the total cost of the transaction via the payment terminal of: 18.15 EUR - Charging Station receives the settlement status from payment terminal - The <i>NotifySettlementRequest</i> is sent to the CSMS around the time the transaction ends, so depending on the <i>TxStopPoint</i> it can be transmitted during or after steps 6 and 7	
8. The Charging Station sends a NotifySettlementRequest	9. The Test System responds with a NotifySettlementResponse with receiptUrl <not omitted>

Tool validations

* Step 2:

Message **SetDefaultTariffResponse**

- **status** must be *Accepted*

* Step 3 and 4 and 5 combined: (First transactionEventRequest with idToken)

Message **TransactionEventRequest** with:

- **idToken** must be *<Not empty>*
- **idToken.type** must be *DirectPayment*
- **transactionInfo.transactionId** must be *<transactionId>*
- **transactionInfo.transactionLimit.maxCost** must be *50.00*

* Step 6 and 7 combined (TransactionEventRequest with eventType Ended):

Message **TransactionEventRequest** with:

- **costDetails.totalCost.currency** must be *EUR*
- **costDetails.totalCost.fixed.exclTax** must be *15.00*
- **costDetails.totalCost.fixed.inclTax** must be *18.15 EUR*
- **costDetails.totalCost.fixed.taxRate.type** must be *MyTax*
- **costDetails.totalCost.fixed.taxRate.tax** must be *21*

* Step 8:

Message **NotifySettlementRequest**

- **transactionId** must be *<transactionId>*
- **pspRef** must be *<Equal to idToken from TransactionEventRequest with triggerReason Authorized or RemoteStart>*
- **status** must be *<not omitted>*
- **settlementTime** must be *<not omitted>*
- **settlementAmount** must be *18.15*
- **receiptUrl** must be *<omitted>*

Post scenario validations:

- The transaction is settled successfully with the payment terminal for 18.15 EUR and the receipt is provided by the CSMS.

TC_C_119_CS: Settlement - is rejected or fails - Failed

Test case name	Settlement - is rejected or fails - Failed
Test case Id	TC_C_119_CS
Use case Id(s)	C22
Requirement(s)	C22.FR.02
System under test	Charging Station
Description	In order to test that Charging Station supports settlement at the end of a transaction where the settlement is rejected or fails.
Purpose	To verify if the Charging Station is able to handle settlement at the end of a transaction where the settlement is rejected or fails.
Prerequisite(s)	- Charging Station has an integrated Payment Terminal - Payment Terminal settlement can be forced to fail

Before (Preparations)
Configuration State: TariffCostCtrlr.Enabled[Tariff] is <i>true</i> TariffCostCtrlr.Enabled[Cost] is <i>true</i> TariffCostCtrlr.Currency is <i>EUR</i> PaymentCtrlr.Enabled is <i>true</i> PaymentCtrlr.AuthorizationAmount is <i>50.00</i> PaymentCtrlr.AuthorizeDirectPayment is <i>false</i> PaymentCtrlr.SettlementByCSMS is <i>false</i> PaymentCtrlr.ReceiptByCSMS is <i>false</i>
Memory State: N/a
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetDefaultTariffResponse	1. The Test System sends a SetDefaultTariffRequest with evseld 0 tariff.tariffId <i>Test System1</i> tariff.currency <i>EUR</i> tariff.fixedFee.taxRates[0].type <i>MyTax</i> tariff.fixedFee.taxRates[0].tax 21 tariff.fixedFee.prices.priceFixed[0] 15.00
3. Execute Reusable State <i>AuthorizedPaymentTerminal</i> (Local)	
4. Execute Reusable State <i>EVConnectedPreSession</i>	
5. Execute Reusable State <i>EnergyTransferStarted</i>	
<u>Manual Action:</u> <i>Simulate payment terminal connection failure (e.g., network failure).</i>	
6. Execute Reusable State <i>StopAuthorized</i> (Remote)	
7. Execute Reusable State <i>ParkingBayUnoccupied</i>	
<u>Note(s):</u> - <i>Charging Station attempts to settle the total cost of the transaction via the payment terminal of: 18.15 EUR</i> - <i>Charging Station receives a Failed settlement status from payment terminal</i> - <i>The NotifySettlementRequest is sent to the CSMS around the time the transaction ends, so depending on the TxStopPoint it can be transmitted during or after steps 6 and 7</i>	
8. The Charging Station sends a NotifySettlementRequest	9. The Test System responds with a NotifySettlementResponse

Tool validations

* Step 2:

Message **SetDefaultTariffResponse**

- **status** must be *Accepted*

* Step 3 and 4 and 5 combined: (First transactionEventRequest with idToken)

Message **TransactionEventRequest** with:

- **idToken** must be *<Not empty>*
- **idToken.type** must be *DirectPayment*
- **transactionInfo.transactionId** must be *<transactionId>*
- **transactionInfo.transactionLimit.maxCost** must be *50.00*

* Step 6 and 7 combined (TransactionEventRequest with eventType Ended):

Message **TransactionEventRequest** with:

- **costDetails.totalCost.currency** must be *EUR*
- **costDetails.totalCost.fixed.exclTax** must be *15.00*
- **costDetails.totalCost.fixed.inclTax** must be *18.15 EUR*
- **costDetails.totalCost.fixed.taxRate.type** must be *MyTax*
- **costDetails.totalCost.fixed.taxRate.tax** must be *21*

* Step 8:

Message **NotifySettlementRequest**

- **transactionId** must be *<transactionId>*
- **pspRef** must be *<Equal to idToken from TransactionEventRequest with triggerReason Authorized or RemoteStart>*
- **status** must be *Failed*
- **settlementTime** must be *<not omitted>*
- **settlementAmount** must be *18.15*
- **receiptUrl** must be *<omitted>*
- **receiptId** must be *<omitted>*

Post scenario validations:

- The transaction settlement failed with the payment terminal for 18.15 EUR.

TC_C_120_CS: Settlement - is rejected or fails - Rejected

Test case name	Settlement - is rejected or fails - Rejected
Test case Id	TC_C_120_CS
Use case Id(s)	C22
Requirement(s)	C22.FR.01
System under test	Charging Station
Description	In order to test that Charging Station supports settlement at the end of a transaction where the settlement is rejected or fails.
Purpose	To verify if the Charging Station is able to handle settlement at the end of a transaction where the settlement is rejected or fails.
Prerequisite(s)	- Charging Station has an integrated Payment Terminal - Payment Terminal can be configured to reject the settlement

Before (Preparations)
<p>Configuration State: TariffCostCtrlr.Enabled[Tariff] is <i>true</i> TariffCostCtrlr.Enabled[Cost] is <i>true</i> TariffCostCtrlr.Currency is <i>EUR</i> PaymentCtrlr.Enabled is <i>true</i> PaymentCtrlr.AuthorizationAmount is <i>50.00</i> PaymentCtrlr.AuthorizeDirectPayment is <i>false</i> PaymentCtrlr.SettlementByCSMS is <i>false</i> PaymentCtrlr.ReceiptByCSMS is <i>false</i></p> <p>Payment Terminal is configured to reject settlement.</p> <p>Memory State: N/a</p> <p>Reusable State(s): N/a</p>

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetDefaultTariffResponse	1. The Test System sends a SetDefaultTariffRequest with evseld 0 tariff.tariffId Test System1 tariff.currency EUR tariff.fixedFee.taxRates[0].type MyTax tariff.fixedFee.taxRates[0].tax 21 tariff.fixedFee.prices.priceFixed[0] 15.00
3. Execute Reusable State <i>AuthorizedPaymentTerminal</i> (Local)	
4. Execute Reusable State <i>EVConnectedPreSession</i>	
5. Execute Reusable State <i>EnergyTransferStarted</i>	
<u>Manual Action</u> : Simulate payment terminal payment rejection for settlement.	
6. Execute Reusable State <i>StopAuthorized</i> (Remote)	
7. Execute Reusable State <i>ParkingBayUnoccupied</i>	
<u>Note(s)</u> : - Charging Station attempts to settle the total cost of the transaction via the payment terminal of: 18.15 EUR - Charging Station receives a Rejected settlement status from payment terminal - The NotifySettlementRequest is sent to the CSMS around the time the transaction ends, so depending on the TxStopPoint it can be transmitted during or after steps 6 and 7	
8. The Charging Station sends a NotifySettlementRequest	9. The Test System responds with a NotifySettlementResponse

Tool validations

* Step 2:

Message **SetDefaultTariffResponse**

- **status** must be *Accepted*

* Step 3 and 4 and 5 combined: (First transactionEventRequest with idToken)

Message **TransactionEventRequest** with:

- **idToken** must be *<Not empty>*
- **idToken.type** must be *DirectPayment*
- **transactionInfo.transactionId** must be *<transactionId>*
- **transactionInfo.transactionLimit.maxCost** must be *50.00*

* Step 6 and 7 combined (TransactionEventRequest with eventType Ended):

Message **TransactionEventRequest** with:

- **costDetails.totalCost.currency** must be *EUR*
- **costDetails.totalCost.fixed.exclTax** must be *15.00*
- **costDetails.totalCost.fixed.inclTax** must be *18.15 EUR*
- **costDetails.totalCost.fixed.taxRate.type** must be *MyTax*
- **costDetails.totalCost.fixed.taxRate.tax** must be *21*

* Step 8:

Message **NotifySettlementRequest**

- **transactionId** must be *<transactionId>*
- **pspRef** must be *<Equal to idToken from TransactionEventRequest with triggerReason Authorized or RemoteStart>*
- **status** must be *Rejected*
- **settlementTime** must be *<not omitted>*
- **settlementAmount** must be *18.15*
- **receiptUrl** must be *<omitted>*
- **receiptId** must be *<omitted>*

Post scenario validations:

- The transaction settlement was rejected by the payment terminal for 18.15 EUR.

TC_C_121_CS: Incremental authorization - increasing enabled

Test case name	Incremental authorization - increasing enabled
Test case Id	TC_C_121_CS
Use case Id(s)	C23
Requirement(s)	C23.FR.01, C23.FR.02
System under test	Charging Station
Description	In order to test that Charging Station supports incremental authorization with increasing enabled.
Purpose	To verify if the Charging Station is able to handle incremental authorization with increasing enabled.
Prerequisite(s)	- Charging Station has an integrated Payment Terminal

Before (Preparations)

Configuration State:

TariffCostCtrlr.Enabled[Tariff] is *true*
TariffCostCtrlr.Enabled[Cost] is *true*
TariffCostCtrlr.Currency is *EUR*
PaymentCtrlr.Enabled is *true*
PaymentCtrlr.AuthorizationAmount is *10.00*
PaymentCtrlr.AuthorizeDirectPayment is *false*
PaymentCtrlr.SettlementByCSMS is *false*
PaymentCtrlr.ReceiptByCSMS is *false*
PaymentCtrlr.IncrementalAuthorizationAmount is *5.00*
PaymentCtrlr.IncrementalAuthorizationThreshold is *2.00*

Memory State:

N/a

Reusable State(s):

N/a

Main (Test scenario)

Charging Station	CSMS
2. The Charging Station responds with a SetDefaultTariffResponse	1. The Test System sends a SetDefaultTariffRequest with evseld 0 tariff.tariffId <i>Test System1</i> tariff.currency <i>EUR</i> tariff.chargingTime.prices[0].priceMinute <i>8.0</i> tariff.chargingTime.taxRates[0].type <i>MyTax</i> tariff.chargingTime.taxRates[0].tax <i>25</i>
3. Execute Reusable State <i>AuthorizedPaymentTerminal</i> (Local)	
4. Execute Reusable State <i>EVConnectedPreSession</i>	
5. Execute Reusable State <i>EnergyTransferStarted</i>	
Note(s): - When cost reaches 8.00 after around 48 seconds since start of the transaction - The Charging Station requests payment terminal to increase the authorization amount to 15.00 EUR	
6. The Charging Station sends a TransactionEventRequest	7. The Test System responds with a TransactionEventResponse
Note(s): - When cost reaches 13.00 after about 78 seconds since start of the transaction - The Charging Station requests payment terminal to increase the authorization amount to 20.00 EUR	
8. The Charging Station sends a TransactionEventRequest	9. The Test System responds with a TransactionEventResponse
Note(s): - When cost reaches 17.00 after about 102 seconds have passed since start of the transaction - The Charging Station requests payment terminal to increase the authorization amount to 25.00 EUR	

Main (Test scenario)	
10. The Charging Station sends a TransactionEventRequest	11. The Test System responds with a TransactionEventResponse

Tool validations
<p>* Step 2: Message SetDefaultTariffResponse - status must be <i>Accepted</i></p> <p>* Step 3 and 4 and 5 combined: (First transactionEventRequest with idToken) Message TransactionEventRequest with: - idToken must be <i><Not empty></i> - idToken.type must be <i>DirectPayment</i> - transactionInfo.transactionId must be <i><transactionId></i> - transactionInfo.transactionLimit.maxCost must be <i>10.00</i></p> <p>* Step 6: Message TransactionEventRequest - eventType must be <i>Updated</i> - transactionInfo.transactionId must be <i><transactionId></i> - transactionInfo.transactionLimit.maxCost must be <i>15.00</i></p> <p>* Step 8: Message TransactionEventRequest - eventType must be <i>Updated</i> - transactionInfo.transactionId must be <i><transactionId></i> - transactionInfo.transactionLimit.maxCost must be <i>20.00</i></p> <p>* Step 10: Message TransactionEventRequest - eventType must be <i>Updated</i> - transactionInfo.transactionId must be <i><transactionId></i> - transactionInfo.transactionLimit.maxCost must be <i>25.00</i></p> <p>Post scenario validations: N/a</p>

TC_C_122_CS: Incremental authorization - increasing disabled

Test case name	Incremental authorization - increasing disabled
Test case Id	TC_C_122_CS
Use case Id(s)	C23
Requirement(s)	C23.FR.01, E16.FR.05, E16.FR.06
System under test	Charging Station
Description	In order to test that Charging Station supports incremental authorization with increasing disabled.
Purpose	To verify if the Charging Station is able to handle incremental authorization with increasing disabled.
Prerequisite(s)	- Charging Station has an integrated Payment Terminal

Before (Preparations)

Configuration State:

TariffCostCtrlr.Enabled[Tariff] is *true*

TariffCostCtrlr.Enabled[Cost] is *true*

TariffCostCtrlr.Currency is *EUR*

PaymentCtrlr.Enabled is *true*

PaymentCtrlr.AuthorizationAmount is *10.00*

PaymentCtrlr.AuthorizeDirectPayment is *false*

PaymentCtrlr.SettlementByCSMS is *false*

PaymentCtrlr.ReceiptByCSMS is *false*

PaymentCtrlr.IncrementalAuthorizationAmount is *0*

PaymentCtrlr.IncrementalAuthorizationThreshold is *2.00*

Memory State:

N/a

Reusable State(s):

N/a

Main (Test scenario)

Charging Station	CSMS
2. The Charging Station responds with a SetDefaultTariffResponse	1. The Test System sends a SetDefaultTariffRequest with evseld 0 tariff.tariffId <i>Test System1</i> tariff.currency <i>EUR</i> tariff.chargingTime.prices[0].priceMinute <i>8.0</i> tariff.chargingTime.taxRates[0].type <i>MyTax</i> tariff.chargingTime.taxRates[0].tax <i>25</i>
3. Execute Reusable State <i>AuthorizedPaymentTerminal</i> (Local)	
4. Execute Reusable State <i>EVConnectedPreSession</i>	
5. Execute Reusable State <i>EnergyTransferStarted</i>	
Note(s): - When <i>10.00</i> is reached after about 60 seconds since start of the transaction - The Charging Station has not increased the authorization amount and will stop charging.	
6. The Charging Station sends a TransactionEventRequest	7. The Test System responds with a TransactionEventResponse

Tool validations

* Step 2:

Message **SetDefaultTariffResponse**

- **status** must be *Accepted*

* Step 3 and 4 and 5 combined: (First transactionEventRequest with idToken)

Message **TransactionEventRequest** with:

- **idToken** must be *<Not empty>*
- **idToken.type** must be *DirectPayment*
- **transactionInfo.transactionId** must be *<transactionId>*
- **transactionInfo.transactionLimit.maxCost** must be *10.00*

* Step 6:

Message: **TransactionEventRequest**

- **eventType** must be *Updated* (or *Ended* if TxStopPoint contains "EnergyTransfer")
- **triggerReason** must be *CostLimitReached*
- **transactionInfo.transactionLimit** must be *<omitted>*
- **transactionInfo.chargingState** must be *SuspendedEVSE* (or *EVConnected* if TxStopPoint contains "EnergyTransfer")

Post scenario validations:

- The Charging Station does not increase the authorization amount when incremental authorization is disabled.

TC_C_123_CS: Ad hoc payment via stand-alone payment terminal - local cost calculation

Test case name	Ad hoc payment via stand-alone payment terminal - local cost calculation
Test case Id	TC_C_123_CS
Use case Id(s)	C24
Requirement(s)	C24.FR.03, C24.FR.04
System under test	Charging Station
Description	In order to test that Charging Station supports ad hoc payment via stand-alone payment terminal with local cost calculation.
Purpose	To verify if the Charging Station is able to handle ad hoc payment via stand-alone payment terminal with local cost calculation.
Prerequisite(s)	- Charging Stations does not have an integrated Payment Terminal. Payment kiosk is used instead.

Before (Preparations)
Configuration State: TariffCostCtrlr.Enabled[Tariff] <i>true</i> TariffCostCtrlr.Enabled[Cost] <i>true</i> TariffCostCtrlr.Currency is <i>EUR</i> PaymentCtrlr.Enabled is <i>false</i> AuthCtrlr.AuthorizeRemoteStart is <i>true</i> (If mutability is ReadWrite)
Memory State: N/a
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetDefaultTariffResponse	1. The Test System sends a SetDefaultTariffRequest with evseld 0 tariff.tariffId <i>Test System1</i> tariff.currency <i>EUR</i> tariff.fixedFee.taxRates[0].type <i>MyTax</i> tariff.fixedFee.taxRates[0].tax 21 tariff.fixedFee.prices.priceFixed[0] 12.45
3. Execute Reusable State <i>AuthorizedPaymentTerminal</i> (Remote) with: (At <i>TransactionEventResponse</i>) - transactionLimit.maxCost 50.00	
4. Execute Reusable State <i>EVConnectedPreSession</i>	
5. Execute Reusable State <i>EnergyTransferStarted</i>	

Tool validations
* Step 2: Message SetDefaultTariffResponse - status must be <i>Accepted</i>
* Step 3 and 4 and 5 combined: (First <i>transactionEventRequest</i> with <i>idToken</i>) Message TransactionEventRequest with: - idToken must be <i><Not empty></i> - idToken.type must be <i>DirectPayment</i> - transactionInfo.transactionId must be <i><transactionId></i> - transactionInfo.transactionLimit.maxCost must be 50.00

Tool validations
Post scenario validations: N/a

TC_C_124_CS: Ad hoc payment via stand-alone payment terminal - central cost calculation

Test case name	Ad hoc payment via stand-alone payment terminal - central cost calculation
Test case Id	TC_C_124_CS
Use case Id(s)	C24
Requirement(s)	C24.FR.03, C24.FR.04
System under test	Charging Station
Description	In order to test that Charging Station supports ad hoc payment via stand-alone payment terminal with central cost calculation.
Purpose	To verify if the Charging Station is able to handle ad hoc payment via stand-alone payment terminal with central cost calculation.
Prerequisite(s)	- Charging Stations does not have an integrated Payment Terminal. Payment kiosk is used instead.

Before (Preparations)
Configuration State: TariffCostCtrlr.Enabled[Tariff] is irrelevant TariffCostCtrlr.Enabled[Cost] <i>false</i> TariffCostCtrlr.Currency is <i>EUR</i> PaymentCtrlr.Enabled is <i>false</i> AuthCtrlr.AuthorizeRemoteStart is <i>true</i> (If mutability is ReadWrite)
Memory State: N/a
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetDefaultTariffResponse	1. The Test System sends a SetDefaultTariffRequest with evseld 0 tariff.tariffId <i>Test System1</i> tariff.currency <i>EUR</i> tariff.fixedFee.taxRates[0].type <i>MyTax</i> tariff.fixedFee.taxRates[0].tax 21 tariff.fixedFee.prices.priceFixed[0] 12.45
3. Execute Reusable State <i>AuthorizedPaymentTerminal</i> (Remote) with: (At the first <i>TransactionEventResponse</i>) - transactionLimit.maxCost 50.00	
4. Execute Reusable State <i>EVConnectedPreSession</i> with: (At the first <i>TransactionEventResponse</i>) - transactionLimit.maxCost 50.00	
5. Execute Reusable State <i>EnergyTransferStarted</i> with: (At the first <i>TransactionEventResponse</i>) - transactionLimit.maxCost 50.00	
Note: - Step 6 and 7 may be executed after the <i>TransactionEventResponse</i> with transactionLimit.maxCost 50.00 has been received. - It is also allowed to transmit the transactionLimit.maxCost 50.00 at the first <i>TransactionEventRequest</i> sent (without <i>triggerReason</i> = <i>LimitSet</i>) after the <i>TransactionEventResponse</i> with transactionLimit.maxCost 50.00 has been received. .	
6. The Charging Station sends a TransactionEventRequest	7. The Test System responds with a TransactionEventResponse

Tool validations
<p>* Step 3 and 4 and 5 combined: (First transactionEventRequest with idToken)</p> <p>Message TransactionEventRequest with:</p> <ul style="list-style-type: none">- idToken must be <i><Not empty></i>- idToken.type must be <i>DirectPayment</i>- transactionInfo.transactionId must be <i><transactionId></i>- transactionInfo.transactionLimit.maxCost must be <i>50.00</i> <p>* Step 6:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none">- triggerReason must be <i>LimitSet</i>- transactionInfo.transactionLimit.maxCost must be <i>50.00</i>
<p>Post scenario validations:</p> <p>N/a</p>

TC_C_127_CS: Ad hoc payment via static or dynamic QR code - no URL parameters

Test case name	Ad hoc payment via static or dynamic QR code - no URL parameters
Test case Id	TC_C_127_CS
Use case Id(s)	C25
Requirement(s)	C25.FR.01, C25.FR.26, C25.FR.27, C25.FR.28
System under test	Charging Station
Description	To provide a static or dynamic QR code with a URL for ad hoc payment
Purpose	To verify if the Charging Station supports ad hoc payments with a url without URL parameters.
Prerequisite(s)	Charging Station supports QR code payment

Before (Preparations)
<p>Configuration State:</p> <p>TariffCostCtrlr.Enabled[Tariff] <i>true</i></p> <p>TariffCostCtrlr.Enabled[Cost] <i>true</i></p> <p>TariffCostCtrlr.Currency is <i>EUR</i></p> <p>AuthCtrlr.AuthorizeRemoteStart is <i>true</i></p> <p>WebPaymentsCtrlr.Enabled is <i>true</i></p> <p>WebPaymentsCtrlr.URLParameters is absent or empty</p> <p>WebPaymentsCtrlr.TOTPVersion is <i>v1</i></p> <p>WebPaymentsCtrlr.ValidityTime is <i>120</i></p> <p>WebPaymentsCtrlr.SharedSecret is <i>12345678</i></p> <p>WebPaymentsCtrlr.Length is <i>8</i></p>
<p>Memory State:</p> <p>N/a</p>
<p>Reusable State(s):</p> <p>N/a</p>

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetDefaultTariffResponse	1. The Test System sends a SetDefaultTariffRequest with evseld 0 tariff.tariffId Test System1 tariff.currency EUR tariff.fixedFee.taxRates[0].type MyTax tariff.fixedFee.taxRates[0].tax 21 tariff.fixedFee.prices.priceFixed[0] 12.45
<u>Note(s):</u> - The Charging Station displays a dynamic QR code according to configuration variables in WebPaymentsCtrlr	
Simulate starting a transaction with an QR code	
4. The Charging Station SHALL respond with a NotifyWebPaymentStartedResponse	3. The Test System sends a NotifyWebPaymentStartedRequest with evseld <configured evseld> timeout 5
5. Execute Reusable State <i>AuthorizedPaymentTerminal</i> (Remote)	
6. Execute Reusable State <i>EVConnectedPreSession</i>	
7. Execute Reusable State <i>EnergyTransferStarted</i>	

Tool validations
<p>* Step 5 and 6 and 7 combined: (First transactionEventRequest with idToken)</p> <p>Message TransactionEventRequest with:</p> <ul style="list-style-type: none">- idToken must be <i><Not empty></i>- idToken.type must be <i>DirectPayment</i>- transactionInfo.transactionId must be <i><transactionId></i>- transactionInfo.transactionLimit must be omitted
<p>Post scenario validations:</p> <p>N/a</p>

TC_C_128_CS: Ad hoc payment via static or dynamic QR code - URL parameter maxTime

Test case name	Ad hoc payment via static or dynamic QR code - URL parameter maxTime
Test case Id	TC_C_128_CS
Use case Id(s)	C25
Requirement(s)	C25.FR.01, C25.FR.03, C25.FR.04
System under test	Charging Station
Description	To provide a static or dynamic QR code with a URL for ad hoc payment
Purpose	To verify if the Charging Station supports ad hoc payments with a url with URL parameters maxTime.
Prerequisite(s)	- Charging Station supports QR codes with a maximum time for the session

Before (Preparations)

Configuration State:

TariffCostCtrlr.Enabled[Tariff] *true*
TariffCostCtrlr.Enabled[Cost] *true*
TariffCostCtrlr.Currency is *EUR*
AuthCtrlr.AuthorizeRemoteStart is *true*
WebPaymentsCtrlr.Enabled is *true*
WebPaymentsCtrlr.URLParameters contains *maxtime*
WebPaymentsCtrlr.TOTPVersion is *v1*
WebPaymentsCtrlr.ValidityTime is *120*
WebPaymentsCtrlr.SharedSecret is *12345678*
WebPaymentsCtrlr.Length is *8*

Memory State:

N/a

Reusable State(s):

N/a

Main (Test scenario)

Charging Station	CSMS
2. The Charging Station responds with a SetDefaultTariffResponse	1. The Test System sends a SetDefaultTariffRequest with evseld 0 tariff.tariffId <i>Test System1</i> tariff.currency <i>EUR</i> tariff.fixedFee.taxRates[0].type <i>MyTax</i> tariff.fixedFee.taxRates[0].tax 21 tariff.fixedFee.prices.priceFixed[0] 12.45
<u>Manual Action:</u> Enter time limit 300 for transaction	
<u>Note(s):</u> - The Charging Station displays a dynamic QR code according to configuration variables in <i>WebPaymentsCtrlr</i>	
Simulate starting a transaction with an QR code	
4. The Charging Station SHALL respond with a NotifyWebPaymentStartedResponse	3. The Test System sends a NotifyWebPaymentStartedRequest with evseld <configured evseld> timeout 5
5. Execute Reusable State <i>AuthorizedPaymentTerminal</i> (Remote) with: (At the first <i>TransactionEventResponse</i>) - transactionLimit.maxTime 300	
6. Execute Reusable State <i>EVConnectedPreSession</i> with: (At the first <i>TransactionEventResponse</i>) - transactionLimit.maxTime 300	

Main (Test scenario)	
<p>7. Execute Reusable State <i>EnergyTransferStarted</i> with: (At the first <i>TransactionEventResponse</i>)</p> <ul style="list-style-type: none"> - transactionLimit.maxTime 300 	
<p><u>Note:</u></p> <ul style="list-style-type: none"> - Step 8 and 9 may be executed after the <i>TransactionEventResponse</i> with transactionLimit.maxTime 300 has been received. - It is also allowed to transmit the transactionLimit.maxTime 300 at the first <i>TransactionEventRequest</i> sent (without <i>triggerReason</i> = <i>LimitSet</i>) after the <i>TransactionEventResponse</i> with transactionLimit.maxTime 300 has been received. . 	
8. The Charging Station sends a TransactionEventRequest	9. The Test System responds with a TransactionEventResponse
Tool validations	
<p>* Step 5 and 6 and 7 combined: (First <i>transactionEventRequest</i> with <i>idToken</i>)</p> <p>Message TransactionEventRequest with:</p> <ul style="list-style-type: none"> - idToken must be <Not empty> - idToken.type must be <i>DirectPayment</i> - transactionInfo.transactionId must be <<i>transactionId</i>> - transactionInfo.transactionLimit.maxTime must be 300 <p>* Step 8:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - triggerReason must be <i>LimitSet</i> - transactionInfo.transactionLimit.maxTime must be 300 	
N/a	
<p>Post scenario validations:</p> <p>N/a</p>	

TC_C_129_CS: Ad hoc payment via static or dynamic QR code - URL parameter maxCost

Test case name	Ad hoc payment via static or dynamic QR code - URL parameter maxCost
Test case Id	TC_C_129_CS
Use case Id(s)	C25
Requirement(s)	C25.FR.01, C25.FR.03, C25.FR.06, C25.FR.58
System under test	Charging Station
Description	To provide a static or dynamic QR code with a URL for ad hoc payment
Purpose	To verify if the Charging Station supports ad hoc payments with a url with URL parameters maxCost.
Prerequisite(s)	- Charging Station supports QR codes with a maximum cost for the session

Before (Preparations)

Configuration State:

TariffCostCtrlr.Enabled[Tariff] *true*
TariffCostCtrlr.Enabled[Cost] *true*
TariffCostCtrlr.Currency is *EUR*
AuthCtrlr.AuthorizeRemoteStart is *true*
WebPaymentsCtrlr.Enabled is *true*
WebPaymentsCtrlr.URLParameters contains *maxcost*
WebPaymentsCtrlr.TOTPVersion is *v1*
WebPaymentsCtrlr.ValidityTime is *120*
WebPaymentsCtrlr.SharedSecret is *12345678*
WebPaymentsCtrlr.Length is *8*

Memory State:

N/a

Reusable State(s):

N/a

Main (Test scenario)

Charging Station	CSMS
2. The Charging Station responds with a SetDefaultTariffResponse	1. The Test System sends a SetDefaultTariffRequest with evseld 0 tariff.tariffId <i>Test System1</i> tariff.currency <i>EUR</i> tariff.fixedFee.taxRates[0].type <i>MyTax</i> tariff.fixedFee.taxRates[0].tax 21 tariff.fixedFee.prices.priceFixed[0] 12.45
<u>Manual Action:</u> Enter cost limit 50.00 for transaction	
<u>Note(s):</u> - The Charging Station displays a dynamic QR code according to configuration variables in <i>WebPaymentsCtrlr</i>	
Simulate starting a transaction with an QR code	
4. The Charging Station SHALL respond with a NotifyWebPaymentStartedResponse	3. The Test System sends a NotifyWebPaymentStartedRequest with evseld <configured evseld> timeout 5
5. Execute Reusable State <i>AuthorizedPaymentTerminal</i> (Remote) with: (At the first <i>TransactionEventResponse</i>) - transactionLimit.maxCost 50.00	
6. Execute Reusable State <i>EVConnectedPreSession</i> with: (At the first <i>TransactionEventResponse</i>) - transactionLimit.maxCost 50.00	

Main (Test scenario)	
<p>7. Execute Reusable State <i>EnergyTransferStarted</i> with: <i>(At the first TransactionEventResponse)</i></p> <ul style="list-style-type: none"> - transactionLimit.maxCost 50.00 	
<p><u>Note:</u></p> <ul style="list-style-type: none"> - Step 8 and 9 may be executed after the TransactionEventResponse with transactionLimit.maxCost 50.00 has been received. - It is also allowed to transmit the transactionLimit.maxCost 50.00 at the first TransactionEventRequest sent (without triggerReason = LimitSet) after the TransactionEventResponse with transactionLimit.maxCost 50.00 has been received. . 	
8. The Charging Station sends a TransactionEventRequest	9. The Test System responds with a TransactionEventResponse
Tool validations	
<p>* Step 5 and 6 and 7 combined: (First transactionEventRequest with idToken)</p> <p>Message TransactionEventRequest with:</p> <ul style="list-style-type: none"> - idToken must be <Not empty> - idToken.type must be <i>DirectPayment</i> - transactionInfo.transactionId must be <transactionId> - transactionInfo.transactionLimit.maxCost must be 50.00 <p>* Step 8:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - triggerReason must be <i>LimitSet</i> - transactionInfo.transactionLimit.maxCost must be 50.00 	
N/a	
<p>Post scenario validations:</p> <p>N/a</p>	

TC_C_130_CS: Ad hoc payment via static or dynamic QR code - URL parameter maxEnergy

Test case name	Ad hoc payment via static or dynamic QR code - URL parameter maxEnergy
Test case Id	TC_C_130_CS
Use case Id(s)	C25
Requirement(s)	C25.FR.01, C25.FR.03, C25.FR.05
System under test	Charging Station
Description	To provide a static or dynamic QR code with a URL for ad hoc payment
Purpose	To verify if the Charging Station supports ad hoc payments with a url with URL parameters maxEnergy.
Prerequisite(s)	- Charging Station supports QR codes with a maximum energy for the session

Before (Preparations)

Configuration State:

TariffCostCtrlr.Enabled[Tariff] *true*
TariffCostCtrlr.Enabled[Cost] *true*
TariffCostCtrlr.Currency is *EUR*
AuthCtrlr.AuthorizeRemoteStart is *true*
WebPaymentsCtrlr.Enabled is *true*
WebPaymentsCtrlr.URLParameters contains *maxenergy*
WebPaymentsCtrlr.TOTPVersion is *v1*
WebPaymentsCtrlr.ValidityTime is *120*
WebPaymentsCtrlr.SharedSecret is *12345678*
WebPaymentsCtrlr.Length is *8*

Memory State:

N/a

Reusable State(s):

N/a

Main (Test scenario)

Charging Station	CSMS
2. The Charging Station responds with a SetDefaultTariffResponse	1. The Test System sends a SetDefaultTariffRequest with evseld 0 tariff.tariffId <i>Test System1</i> tariff.currency <i>EUR</i> tariff.fixedFee.taxRates[0].type <i>MyTax</i> tariff.fixedFee.taxRates[0].tax 21 tariff.fixedFee.prices.priceFixed[0] 12.45
<u>Manual Action:</u> Enter energy limit 20000 kWh for transaction	
<u>Note(s):</u> - The Charging Station displays a dynamic QR code according to configuration variables in <i>WebPaymentsCtrlr</i>	
Simulate starting a transaction with an QR code	
4. The Charging Station SHALL respond with a NotifyWebPaymentStartedResponse	3. The Test System sends a NotifyWebPaymentStartedRequest with evseld <configured evseld> timeout 5
5. Execute Reusable State <i>AuthorizedPaymentTerminal</i> (Remote) with: (At the first <i>TransactionEventResponse</i>) - transactionLimit.maxEnergy 20000	
6. Execute Reusable State <i>EVConnectedPreSession</i> with: (At the first <i>TransactionEventResponse</i>) - transactionLimit.maxEnergy 20000	

Main (Test scenario)	
<p>7. Execute Reusable State <i>EnergyTransferStarted</i> with: (At the first <i>TransactionEventResponse</i>) - transactionLimit.maxEnergy 20000</p>	
<p><u>Note:</u> - Step 8 and 9 may be executed after the <i>TransactionEventResponse</i> with transactionLimit.maxEnergy 20000 has been received. - It is also allowed to transmit the transactionLimit.maxEnergy 20000 at the first <i>TransactionEventRequest</i> sent (without <i>triggerReason</i> = <i>LimitSet</i>) after the <i>TransactionEventResponse</i> with transactionLimit.maxEnergy 20000 has been received. .</p>	
8. The Charging Station sends a TransactionEventRequest	9. The Test System responds with a TransactionEventResponse
Tool validations	
<p>* Step 5 and 6 and 7 combined: (First <i>transactionEventRequest</i> with <i>idToken</i>) Message TransactionEventRequest with: - idToken must be <Not empty> - idToken.type must be <i>DirectPayment</i> - transactionInfo.transactionId must be <<i>transactionId</i>> - transactionInfo.transactionLimit.maxEnergy is 20000</p> <p>* Step 8: Message: TransactionEventRequest - triggerReason must be <i>LimitSet</i> - transactionInfo.transactionLimit.maxEnergy is 20000</p>	
N/a	
<p>Post scenario validations: N/a</p>	

D Local Authorization List Management

TC_D_01_CS: Send Local Authorization List - Full

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 <i>true</i> (If implemented) *Configured versionNumber > 0
Memory State: N/a
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SendLocalListResponse	1. The Test System sends a SendLocalListRequest with - updateType <i>Full</i> - versionNumber <Configured <i>versionNumber</i> > - localAuthorizationList[0].idToken.idToken <Configured <i>valid_idtoken_idtoken</i> > - localAuthorizationList[0].idToken.type <Configured <i>valid_idtoken_type</i> >
4. The Charging Station responds with a GetLocalListVersionResponse	3. The Test System sends a GetLocalListVersionRequest

Tool validations
* Step 2: Message SendLocalListResponse - status <i>Accepted</i> * Step 4: Message GetLocalListVersionResponse - versionNumber <Equal to version sent in step 1>
Post scenario validations: - N/a

TC_D_02_CS: Send Local Authorization List - Differential Update

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 <i>true</i> (If implemented) *Configured versionNumber > 0
Memory State: <i>IdTokenLocalAuthList</i> for <Configured valid idtoken fields>
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SendLocalListResponse	1. The Test System sends a SendLocalListRequest with - updateType <i>Differential</i> - versionNumber <Configured versionNumber + 1> - localAuthorizationList[0].idToken.idToken <Configured <i>valid_idtoken_idtoken2</i> > - localAuthorizationList[0].idToken.type <Configured <i>valid_idtoken_type2</i> >
<u>Note(s):</u> The message send by Test System is within <i>ItemsPerMessageSendLocalList</i> AND <i>BytesPerMessageSendLocalList</i> range.	
4. The Charging Station responds with a GetLocalListVersionResponse	3. The Test System sends a GetLocalListVersionRequest

Tool validations
* Step 2: Message SendLocalListResponse - status <i>Accepted</i> * Step 4: Message GetLocalListVersionResponse - versionNumber <Equal to version send in step 1>
Post scenario validations: - N/a

TC_D_03_CS: Send Local Authorization List - Differential Remove

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 <i>true</i> (If implemented) *Configured versionNumber > 0
Memory State: <i>IdTokenLocalAuthList</i> for <Configured valid idtoken fields>
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SendLocalListResponse	1. The Test System sends a SendLocalListRequest with - updateType <i>Differential</i> - versionNumber <Configured versionNumber + 1> - localAuthorizationList <Contains <i>AuthorizationData</i> elements without <i>idTokenInfo</i> >
<u>Note(s):</u> The message send by Test System is within <i>ItemsPerMessageSendLocalList</i> AND <i>BytesPerMessageSendLocalList</i> range.	
4. The Charging Station responds with a GetLocalListVersionResponse	3. The Test System sends a GetLocalListVersionRequest

Tool validations
* Step 2: Message SendLocalListResponse - status <i>Accepted</i> * Step 3: Message GetLocalListVersionResponse - versionNumber <Equal to version sent in step 1>
Post scenario validations: - N/a

TC_D_04_CS: Send Local Authorization List - Full with empty list

Test case name	Send Local Authorization List - Full with empty 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 <i>true</i> (If implemented) *Configured <i>versionNumber</i> > 0
Memory State: <i>IdTokenLocalAuthList</i> for <Configured valid idtoken fields>
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SendLocalListResponse	1. The Test System sends a SendLocalListRequest with - updateType <i>Full</i> - versionNumber <Configured <i>versionNumber</i> > - localAuthorizationList <Empty>
4. The Charging Station responds with a GetLocalListVersionResponse	3. The Test System sends a GetLocalListVersionRequest

Tool validations
* Step 2: Message SendLocalListResponse - status <i>Accepted</i> * Step 4: Message GetLocalListVersionResponse - versionNumber <Configured <i>versionNumber</i> >
Post scenario validations: - N/a

TC_D_05_CS: Send Local Authorization List - Differential with empty list

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 <i>true</i> (If implemented) *Configured versionNumber > 0
Memory State: <i>IdTokenLocalAuthList</i> for <Configured valid idtoken fields>
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SendLocalListResponse	1. The Test System sends a SendLocalListRequest with - updateType <i>Differential</i> - versionNumber <Configured versionNumber + 1> - localAuthorizationList <Empty>
4. The Charging Station responds with a GetLocalListVersionResponse	3. The Test System sends a GetLocalListVersionRequest

Tool validations
* Step 2: Message SendLocalListResponse - status <i>Accepted</i> * Step 4: Message GetLocalListVersionResponse - versionNumber <Equal to the version send in step 1>
Post scenario validations: - N/a

TC_D_06_CS: Send Local Authorization List - VersionMismatch

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 <i>true</i> (If implemented) *Configured versionNumber > 1
Memory State: <i>IdTokenLocalAuthList</i> for <Configured valid idtoken fields>
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SendLocalListResponse	1. The Test System sends a SendLocalListRequest with - updateType <i>Differential</i> - versionNumber <Configured <i>versionNumber</i> - 1> - localAuthorizationList <Not Empty>
4. The Charging Station responds with a GetLocalListVersionResponse	3. The Test System sends a GetLocalListVersionRequest

Tool validations
* Step 2: Message SendLocalListResponse - status <i>VersionMismatch</i> * Step 4: Message GetLocalListVersionResponse - versionNumber <Configured <i>versionNumber</i> >
Post scenario validations: - N/a

TC_D_07_CS: Send Local Authorization List - Persistent over reboot

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 <i>true</i> (If implemented) *Configured <i>versionNumber</i> > 0
Memory State: <i>IdTokenLocalAuthList</i> for <Configured valid idtoken fields>
Reusable State: <i>Booted</i>

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a GetLocalListVersionResponse	1. The Test System sends a GetLocalListVersionRequest

Tool validations
* Step 2: Message GetLocalListVersionResponse - versionNumber <Configured <i>versionNumber</i> >
Post scenario validations: - N/a

TC_D_08_CS: Get Local List Version - Success

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 <code>GetLocalListVersionRequest</code> .
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 <i>true</i> (If implemented) *Configured <code>versionNumber</code> > 0
Memory State: <i>IdTokenLocalAuthList</i> for <Configured valid idtoken fields>
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a GetLocalListVersionResponse	1. The Test System sends a GetLocalListVersionRequest

Tool validations
* Step 2: Message GetLocalListVersionResponse - versionNumber <Configured <code>versionNumber</code> >
Post scenario validations: - N/a

TC_D_10_CS: Get Local List Version - Function disabled

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 <i>false</i> (If implemented)
Memory State: N/a
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a GetLocalListVersionResponse	1. The Test System sends a GetLocalListVersionRequest

Tool validations
* Step 2: Message GetLocalListVersionResponse - versionNumber 0
Post scenario validations: - N/a

E Transactions

TC_E_01_CS: Start transaction options - PowerPathClosed

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 <i>ReadOnly</i> AND (the value <i>PowerPathClosed</i> is NOT set OR (<i>ParkingBayOccupancy</i> OR <i>EVConnected</i> OR <i>Authorized</i> OR <i>DataSigned</i>), is set). - If the mutability of TxStartPoint is <i>ReadWrite</i> , then the value <i>PowerPathClosed</i> must be supported.

Before (Preparations)
Configuration State: If the mutability of TxStartPoint is <i>ReadWrite</i> then TxStartPoint contains <i>PowerPathClosed</i>
Memory State: N/a
Reusable State(s): State is <i>Authorized</i>

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>EVConnectedPreSession</i>	
2. Execute Reusable State <i>EnergyTransferStarted</i>	

Tool validations
N/a
Post scenario validations: N/a

TC_E_02_CS: Start transaction options - EnergyTransfer

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)	<p>- The Charging Station does NOT have the following configuration; The mutability of TxStartPoint is <i>ReadOnly</i> AND (the value <i>EnergyTransfer</i> is NOT set OR (<i>ParkingBayOccupancy</i> OR <i>EVConnected</i> OR <i>Authorized</i> OR <i>DataSigned</i> OR <i>PowerPathClosed</i>), is set).</p> <p>- If the mutability of TxStartPoint is <i>ReadWrite</i>, then the value <i>EnergyTransfer</i> must be supported.</p>

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
<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 Test System responds accordingly.
3. The Charging Station sends a TransactionEventRequest	4. The Test System 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 Test System 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

TC_E_09_CS: Start transaction options - EVConnected

Test case name	Start transaction options - EVConnected
Test case Id	TC_E_09_CS
Use case Id(s)	E01(S2)
Requirement(s)	E01.FR.02, E01.FR.07, E01.FR.10, E01.FR.15, E01.FR.16
System under test	Charging Station
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the start options.
Purpose	To verify if the Charging Station starts a transaction when the EV and EVSE are connected and it has been configured to do so.
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station does NOT have the following configuration; The mutability of TxStartPoint is <i>ReadOnly</i> AND (the value <i>EVConnected</i> is NOT set OR <i>ParkingBayOccupancy</i> is set). - If the mutability of TxStartPoint is <i>ReadWrite</i>, then the value <i>EVConnected</i> must be supported.

Before (Preparations)
Configuration State: If the mutability of TxStartPoint is <i>ReadWrite</i> then TxStartPoint contains <i>EVConnected</i>
Memory State: N/a
Reusable State(s): State is <i>ParkingBayOccupied</i>

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>EVConnectedPreSession</i>	
2. Execute Reusable State <i>Authorized</i>	
3. Execute Reusable State <i>EnergyTransferStarted</i>	

Tool validations
N/a
Post scenario validations: N/a

TC_E_10_CS: Start transaction options - Authorized - Local

Test case name	Start transaction options - Authorized - Local
Test case Id	TC_E_10_CS
Use case Id(s)	E01(S3) AND (C01 OR C02 OR C04 OR C06)
Requirement(s)	E01.FR.03, E01.FR.07, E01.FR.10, E01.FR.15, E01.FR.16 AND C01.FR.02, C02.FR.01, C06.FR.02
System under test	Charging Station
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the start options.
Purpose	To verify if the Charging Station starts a transaction when the EV and EVSE are connected and it has been configured to do so.
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station supports at least one authorization method described at the following Use cases; C01, C02, C04, C06. - The Charging Station does NOT have the following configuration; The mutability of TxStartPoint is <i>ReadOnly</i> AND the value <i>Authorized</i> is NOT set. - If the mutability of TxStartPoint is <i>ReadWrite</i>, then the value <i>Authorized</i> 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
1. Execute Reusable State <i>Authorized</i> (Local)	
2. Execute Reusable State <i>EVConnectedPreSession</i>	
3. Execute Reusable State <i>EnergyTransferStarted</i>	

Tool validations

N/a

Post scenario validations:

N/a

TC_E_13_CS: Start transaction options - Authorized - Remote

Test case name	Start transaction options - Authorized - Remote
Test case Id	TC_E_13_CS
Use case Id(s)	E01(S3) AND F02
Requirement(s)	E01.FR.03 AND F01.FR.03, F01.FR.04, F01.FR.06, F01.FR.19, F02.FR.01
System under test	Charging Station
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the start options.
Purpose	To verify if the Charging Station starts a transaction when the EV and EVSE are connected and it has been configured to do so.
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station does NOT have the following configuration; The mutability of TxStartPoint is <i>ReadOnly</i> AND the value <i>Authorized</i> is NOT set. - If the mutability of TxStartPoint is <i>ReadWrite</i>, then the value <i>Authorized</i> 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
1. Execute Reusable State <i>Authorized</i> (Remote)	
2. Execute Reusable State <i>EVConnectedPreSession</i>	
3. Execute Reusable State <i>EnergyTransferStarted</i>	

Tool validations

N/a

Post scenario validations:

N/a

TC_E_11_CS: Start transaction options - DataSigned

Test case name	Start transaction options - DataSigned
Test case Id	TC_E_11_CS
Use case Id(s)	E01(S4)
Requirement(s)	E01.FR.04, E01.FR.07, E01.FR.10, E01.FR.15, E01.FR.16
System under test	Charging Station
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the start options.
Purpose	To verify if the Charging Station starts a transaction when the EV and EVSE are connected and it has been configured to do so.
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station does NOT have the following configuration; The mutability of TxStartPoint is <i>ReadOnly</i> AND (the value <i>DataSigned</i> is NOT set OR (<i>ParkingBayOccupancy</i> OR <i>EVConnected</i> OR <i>Authorized</i>), is set). - If the mutability of TxStartPoint is <i>ReadWrite</i>, then the value <i>DataSigned</i> must be supported.

Before (Preparations)
Configuration State: If the mutability of TxStartPoint is <i>ReadWrite</i> then TxStartPoint contains <i>DataSigned</i> SampledDataCtrlr.SignReadings is <i>true</i>
Memory State: N/a
Reusable State(s): State is <i>Authorized</i>

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 Test System responds accordingly.
3. The Charging Station sends a TransactionEventRequest	4. The Test System responds with a TransactionEventResponse
5. The Charging Station sends a TransactionEventRequest	6. The Test System responds with a TransactionEventResponse

Tool validations
<p>* Step 1:</p> <p>Message: StatusNotificationRequest</p> <ul style="list-style-type: none"> - connectorStatus must be <i>Occupied</i> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none"> - eventData[0].trigger must be <i>Delta</i> - eventData[0].actualValue must be <i>Occupied</i> - eventData[0].component.name must be <i>Connector</i> - eventData[0].variable.name must be <i>AvailabilityState</i> <p>* Step 3:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - eventType must be <i>Started</i> - If the Test System is configured to start transactions using a RequestStartTransactionRequest message then triggerReason must be <i>RemoteStart</i> or <i>SignedDataReceived</i> <p>Else triggerReason must be <i>SignedDataReceived</i></p> <ul style="list-style-type: none"> - 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: <p>sampledValue.context is <i>Transaction.Begin</i></p> <p>sampledValue.signedMeterValue.encodingMethod is not omitted</p> <p>sampledValue.signedMeterValue.publicKey is not omitted</p> <p>sampledValue.signedMeterValue.signedMeterData is not omitted</p> <p>sampledValue.signedMeterValue.signingMethod is not omitted</p> <p>* Step 5:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - eventType must be <i>Updated</i> - triggerReason must be <i>ChargingStateChanged</i> - transactionInfo.chargingState must be <i>Charging</i>
<p>Post scenario validations:</p> <p>N/a</p>

TC_E_12_CS: Start transaction options - ParkingBayOccupied

Test case name	Start transaction options - ParkingBayOccupied
Test case Id	TC_E_12_CS
Use case Id(s)	E01(S1)
Requirement(s)	E01.FR.01, E01.FR.07, E01.FR.10, E01.FR.15, E01.FR.16
System under test	Charging Station
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the start options.
Purpose	To verify if the Charging Station starts a transaction when the EV and EVSE are connected and it has been configured to do so.
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station does NOT have the following configuration; The mutability of TxStartPoint is <i>ReadOnly</i> AND the value <i>ParkingBayOccupancy</i> is NOT set. - If the mutability of TxStartPoint is <i>ReadWrite</i>, then the value <i>ParkingBayOccupancy</i> must be supported.

Before (Preparations)
Configuration State: If the mutability of TxStartPoint is <i>ReadWrite</i> then TxStartPoint contains <i>ParkingBayOccupancy</i>
Memory State: N/a
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>ParkingBayOccupied</i>	
2. Execute Reusable State <i>Authorized</i>	
3. Execute Reusable State <i>EVConnectedPreSession</i>	
4. Execute Reusable State <i>EnergyTransferStarted</i>	

Tool validations
N/a
Post scenario validations: N/a

TC_E_16_CS: Stop transaction options - Deauthorized - Invalid idToken

Test case name	Stop transaction options - Deauthorized - Invalid idToken
Test case Id	TC_E_16_CS
Use case Id(s)	E06(S3)
Requirement(s)	E06.FR.04, E06.FR.15, C15.FR.04
System under test	Charging Station
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the start options.
Purpose	To verify if the Charging Station stops a transaction when the transaction gets deauthorized by the status from the idTokenInfo at a TransactionEventResponse message and it has been configured to do so.
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station does NOT have the following configuration; The mutability of TxStopPoint is <i>ReadOnly</i> AND the value <i>Authorized</i> OR <i>PowerPathClosed</i> is NOT set OR (<i>EnergyTransfer</i> OR <i>DataSigned</i> is set). - If the mutability of TxStopPoint is <i>ReadWrite</i>, then the value <i>Authorized</i> OR <i>PowerPathClosed</i> must be supported. - The Charging Station supports local start/stop transaction. - The Charging Station supports authorization methods other than <i>NoAuthorization</i>

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 <u>Note(s):</u> - The Charging Station will empty its Transaction message queue. This will contain one or more TransactionEventRequest messages	2. The Test System responds with a TransactionEventResponse <u>Note(s):</u> - The Test System will respond to the TransactionEventRequest containing the idToken, with idtokenInfo.status <i>Invalid</i>
3. The Charging Stations sends a TransactionEventRequest <u>Note(s):</u> - After having emptied its queue, the Charging Station will send a TransactionEventRequest in which it reports it deauthorizes the transaction.	4. The Test System responds with a TransactionEventResponse

Tool validations
<div>* Step 1: Message: TransactionEventRequest - offline must be <i>true</i></div> <div>* Step 3: Message: TransactionEventRequest - eventType must be <i>Ended</i> - triggerReason must be <i>Deauthorized</i> - transactionInfo.stoppedReason is <i>DeAuthorized</i></div>
<div>Post scenario validations: N/a</div>

TC_E_17_CS: Stop transaction options - Deauthorized - EV side disconnect

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)	This testcase is applicable if the value Authorized is a supported value for TxStopPoint AND EVConnected, PowerPathClosed and EnergyTransfer must not be set as TxStopPoint AND StopTxOnEVSideDisconnect true must be a supported value.

Before (Preparations)

Configuration State:

If the mutability of **TxStopPoint** is *ReadWrite* then **TxStopPoint** contains *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 (Test scenario)

Charging Station	CSMS
<u>Manual Action:</u> Disconnect the EV and EVSE on EV side (EVSE loses connection with EV).	
1. The Charging Station sends a TransactionEventRequest	2. The Test System responds with a TransactionEventResponse
<u>Manual Action:</u> Present the <i>IdToken</i> that was used to start the transaction.	
<u>Note(s):</u> - This manual action needs to be executed when the Charging Station has a detachable cable on the Charging Station side AND <i>UnlockOnEVSideDisconnect</i> is set to <i>false</i> .	
<u>Manual Action:</u> Disconnect the EV and EVSE on Charging Station side.	
<u>Note(s):</u> - 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 Test System responds accordingly.

Tool validations
<div>* Step 1: Message: TransactionEventRequest<ul style="list-style-type: none">- triggerReason must be <i>EVCommunicationLost</i>- transactionInfo.chargingState must be <i>Idle</i>- transactionInfo.stoppedReason must be <i>EVDisconnected</i>- eventType must be <i>Ended</i></div> <div>* Step 3: Message: StatusNotificationRequest<ul style="list-style-type: none">- connectorStatus <i>Available</i></div> <div>Message: NotifyEventRequest<ul style="list-style-type: none">- eventData[0].trigger <i>Delta</i>- eventData[0].actualValue <i>"Available"</i>- eventData[0].component.name <i>"Connector"</i>- eventData[0].variable.name <i>"AvailabilityState"</i></div>
<div>Post scenario validations: N/a</div>

TC_E_39_CS: Stop transaction options - Deauthorized - timeout

Test case name	Stop transaction options - Deauthorized - timeout
Test case Id	TC_E_39_CS
Use case Id(s)	E06(S3)
Requirement(s)	E06.FR.04, E06.FR.15
System under test	Charging Station
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the start options.
Purpose	To verify if the Charging Station stops a transaction when the transaction gets deauthorized because the cable was not plugged in within the Configured durationout and it has been configured to do so.
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station does NOT have the following configuration; The mutability of TxStopPoint is <i>ReadOnly</i> AND the value <i>Authorized</i> is NOT set. - If the mutability of TxStopPoint is <i>ReadWrite</i>, then the value <i>Authorized</i> must be supported.

Before (Preparations)

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 <u>Note(s):</u> - This step needs to be executed after the <i><Configured ev_connection_timeout></i> expires, if the transaction has been started. So in the case TxStartPoint contains <i>ParkingBayOccupancy</i> OR <i>Authorized</i>	2. The Test System responds with a TransactionEventResponse
<u>Note(s):</u> Step 1 and 2 are optional and will only be expected when the <i>TxStartPoint</i> is set to <i>ParkingBayOccupancy</i> or <i>Authorized</i> . Optionally the Charging Station can send a <i>StatusNotificationRequest</i> or <i>NotifyEventRequest</i> with status <i>Available</i> .	
<u>Manual Action:</u> Connect the EV and EVSE on EV side.	
<u>Manual Action:</u> Connect the EV and EVSE on EVSE side.	
3. The Charging Station notifies the CSMS about the status change of the connector.	4. The Test System responds accordingly.
5. The Charging Station sends a TransactionEventRequest <u>Note(s):</u> - This step needs to be executed after the <i><Configured ev_connection_timeout></i> expires, if the transaction has been started. So in the case TxStartPoint contains <i>ParkingBayOccupancy</i> OR <i>Authorized</i>	6. The Test System responds with a TransactionEventResponse
<u>Note(s):</u> Charging Station is allowed to sent a <i>TransactionEventRequest</i> for the <i>cableplugin</i> event when this is applicable, but should not start charging.	

Tool validations
<p>* Step 1:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none">- triggerReason must be <i>EVConnectTimeout</i>- eventType must be <i>Updated</i> if TxStartPoint is <i>ParkingBayOccupancy</i>, else <i>Ended</i>- transactionInfo.stoppedReason must be <i>Timeout</i> <p>* Step 3:</p> <p>Message: StatusNotificationRequest</p> <ul style="list-style-type: none">- connectorStatus must be <i>Occupied</i> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none">- eventData[0].trigger must be <i>Delta</i>- eventData[0].actualValue must be <i>Occupied</i>- eventData[0].component.name must be <i>Connector</i>- eventData[0].variable.name must be <i>AvailabilityState</i> <p>* Step 5:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none">- triggerReason can only be <i>CablePluggedIn</i>- transactionInfo.chargingState should not be <i>Charging</i>- eventType must be <i>Updated</i> if TxStartPoint is <i>ParkingBayOccupancy</i>, else <i>Ended</i>
<p>Post scenario validations:</p> <p>N/a</p>

TC_E_03_CS: Local start transaction - Cable plugin first - Success

Test case name	Local start transaction - Cable plugin first - Success
Test case Id	TC_E_03_CS
Use case Id(s)	E02 AND (C01 OR C02 OR C04 OR C06)
Requirement(s)	E02.FR.01, E02.FR.05, E02.FR.06, E02.FR.07, E02.FR.13, E02.FR.15, E02.FR.16, E02.FR.17, E01.FR.16 AND C01.FR.02, C02.FR.01, C06.FR.02
System under test	Charging Station
Description	OCPP 2.x.x allows an EV driver to either first connect the EV and EVSE OR present a form of identification. Both sequences will result in being able to charge.
Purpose	To verify if the Charging Station is able to start a charging session when the EV driver first connects the EV and EVSE, before authorization.
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station supports at least one authorization method described at the following Use cases; C01, C02, C04, C06. - The Charging Station does NOT have a cable lock, which prevents the EV driver to connect the EV and EVSE before authorization.

Before (Preparations)

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 <i>Authorized</i> (local)	
2. Execute Reusable State <i>EnergyTransferStarted</i>	

Tool validations

N/a

Post scenario validations:

N/a

TC_E_04_CS: Local start transaction - Authorization first - Success

Test case name	Local start transaction - Authorization first - Success
Test case Id	TC_E_04_CS
Use case Id(s)	E03 AND (C01 OR C02 OR C04 OR C06)
Requirement(s)	E03.FR.01, E03.FR.06, E03.FR.12, E01.FR.16 AND C01.FR.02, C02.FR.01, C06.FR.02
System under test	Charging Station
Description	OCPP 2.x.x allows an EV driver to either first connect the EV and EVSE OR present a form of identification. Both sequences will result in being able to charge.
Purpose	To verify if the Charging Station is able to start a charging session when the EV driver first presents a form of identification, before connecting the EV and EVSE.
Prerequisite(s)	The Charging Station supports at least one authorization method described at the following Use cases; C01, C02, C04, C06.

Before (Preparations)
Configuration State: AuthCtrlr.AuthEnabled is <i>true</i> (If implemented AND ReadWrite) AuthCtrlr.DisableRemoteAuthorization is <i>false</i> (If implemented)
Memory State: N/a
Reusable State(s): State is <i>ParkingBayOccupied</i> (Optional state)

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>Authorized</i> (local)	
2. Execute Reusable State <i>EnergyTransferStarted</i>	

Tool validations
N/a
Post scenario validations: N/a

TC_E_05_CS: Local start transaction - Authorization first - Cable plugin timeout

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 <u>Note(s):</u> - 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 <i>ParkingBayOccupancy</i> OR <i>Authorized</i>	2. The Test System responds with a TransactionEventResponse
<u>Note(s):</u> - This step is only executed if TxStartPoint is <i>ParkingBayOccupancy</i> or <i>Authorized</i> - Optionally the Charging Station can send a <i>StatusNotificationRequest</i> or <i>NotifyEventRequest</i> with status <i>Available</i>	
3. Execute Reusable State <i>Authorized</i> (local) <u>Note(s):</u> - This step is executed to verify if the EVSE is actually ready to start another charging session.	
4. Execute Reusable State <i>EnergyTransferStarted</i>	

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

TC_E_38_CS: Local start transaction - EV not ready

Test case name	Local start transaction - EV not ready
Test case Id	TC_E_38_CS
Use case Id(s)	E03
Requirement(s)	N/a
System under test	Charging Station
Description	OCPP 2.x.x allows an EV driver to either first connect the EV and EVSE OR present a form of identification. Both sequences will result in being able to charge.
Purpose	To verify if the Charging Station is able to handle and report if an EV is not ready to start the energy transfer (yet).
Prerequisite(s)	TxStartPoint should not be EnergyTransfer

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>Authorized</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action</u> : 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 Test System responds with a TransactionEventResponse

Tool validations
* Step 2: Message: TransactionEventRequest - triggerReason must be <i>ChargingStateChanged</i> - transactionInfo.chargingState should be <i>SuspendedEV</i>
Post scenario validations: N/a

TC_E_52_CS: Local start transaction - Authorization first - DisableRemoteAuthorization

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)	<ul style="list-style-type: none"> - The Charging Station supports Authorization Cache or the Local Authorization List - The Charging Station supports one of the following local authorization methods; RFID ISO14443 / RFID ISO15693 / KeyCode / MacAddress - AuthCtrl.DisableRemoteAuthorization is implemented.

Before (Preparations)
Configuration State: AuthCtrl.Enabled is <i>true</i> (If implemented) AuthCtrl.DisableRemoteAuthorization is <i>true</i>
Memory State: None of the configured valid IdTokens is present in Authorization Cache or Local Authorization List.
Reusable State(s): State is <i>ParkingBayOccupied</i> (Optional state)

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action</u> : 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	
2. Execute Reusable State <i>EVConnectedPreSession</i>	
3. The Charging Station does NOT start charging.	

Tool validations
* Step 1: Check that Charging Station does NOT send an AuthorizeRequest and authorization is refused.
Post scenario validations: - N/a

TC_E_06_CS: Local Stop Transaction - Accepted

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 <i>true</i> (If implemented AND ReadWrite) AuthCtrlr.DisableRemoteAuthorization is <i>false</i> (If implemented)
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>StopAuthorized</i> (local)	
2. Execute Reusable State <i>EVConnectedPostSession</i>	
3. Execute Reusable State <i>EVDisconnected</i>	
4. Execute Reusable State <i>ParkingBayUnoccupied</i>	

Tool validations
N/a
Post scenario validations: N/a

TC_E_07_CS: Stop transaction options - PowerPathClosed - Local stop

Test case name	Stop transaction options - PowerPathClosed - Local stop
Test case Id	TC_E_07_CS
Use case Id(s)	E06(S5)
Requirement(s)	E06.FR.06, E06.FR.15
System under test	Charging Station
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the stop options.
Purpose	To verify if the Charging Station stops a transaction when it is locally stopped by an EV driver and TxStopPoint contains <i>PowerPathClosed</i> .
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station does NOT have the following configuration; The mutability of TxStopPoint is <i>ReadOnly</i> AND (the value <i>PowerPathClosed</i> is NOT set OR <i>Authorized</i> is set). - If the mutability of TxStopPoint is <i>ReadWrite</i>, then the value <i>PowerPathClosed</i> must be supported.

Before (Preparations)
Configuration State: TxStopPoint contains <i>PowerPathClosed</i>
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action</u> : Present <i>IdToken</i> to stop charging session.	
1. Execute Reusable State <i>StopAuthorized</i>	

Tool validations
N/a
Post scenario validations: N/a

TC_E_35_CS: Stop transaction options - PowerPathClosed - Remote stop

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 <i>PowerPathClosed</i> .
Prerequisite(s)	- The Charging Station does NOT have the following configuration; The mutability of TxStopPoint is <i>ReadOnly</i> AND (the value <i>PowerPathClosed</i> is NOT set OR <i>Authorized</i> is set). - If the mutability of TxStopPoint is <i>ReadWrite</i> , then the value <i>PowerPathClosed</i> must be supported.

Before (Preparations)

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 Test System sends a RequestStopTransactionRequest with transactionId <transactionId provided by the Charging Station in <i>TransactionEventRequest</i> >
3. The Charging Station sends a TransactionEventRequest	4. The Test System responds with a TransactionEventResponse
<u>Note(s)</u> : The charging Station can stop a transaction in more than one <i>TransactionEventRequest</i> .	

Tool validations

* Step 2:

Message: **RequestStopTransactionResponse**

- **status** must be *Accepted*

* Step 3:

Message: **TransactionEventRequest**

- **triggerReason** must be *RemoteStop* (for one of the *TransactionEventRequests*)

- **transactionInfo.stoppedReason** must be *Remote* (for the last *TransactionEventRequest*)

- **eventType** must be *Ended* (for the last *TransactionEventRequest*)

Post scenario validations:

N/a

TC_E_37_CS: Stop transaction options - PowerPathClosed - EV side disconnect

Test case name	Stop transaction options - PowerPathClosed - EV side disconnect
Test case Id	TC_E_37_CS
Use case Id(s)	E06(S5)
Requirement(s)	E06.FR.06, E06.FR.15
System under test	Charging Station
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the stop options.
Purpose	To verify if the Charging Station stops a transaction when the EV and the EVSE get disconnected and TxStopPoint contains <i>PowerPathClosed</i> .
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station does NOT have the following configuration; The mutability of TxStopPoint is <i>ReadOnly</i> AND (the value <i>PowerPathClosed</i> is NOT set OR (<i>EnergyTransfer</i> OR <i>EVConnected</i> OR <i>DataSigned</i> is set)). - If the mutability of TxStopPoint is <i>ReadWrite</i>, then the value <i>PowerPathClosed</i> must be supported.

Before (Preparations)
Configuration State: TxStopPoint contains <i>PowerPathClosed</i> StopTxOnEVSideDisconnect is <i>false</i> (If mutability is <i>ReadWrite</i>)
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferSuspended</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action:</u> Disconnect the EV and EVSE on EV side (EVSE loses connection with EV).	
1. The Charging Station sends a TransactionEventRequest	2. The Test System responds with a TransactionEventResponse

Tool validations
* Step 1: Message: TransactionEventRequest - triggerReason must be <i>EVCommunicationLost</i> - transactionInfo.chargingState must be <i>Idle</i> - transactionInfo.stoppedReason must be <i>EVDDisconnected</i> or <i>StoppedByEV</i> (preferred value) - eventType must be <i>Ended</i>
Post scenario validations: N/a

TC_E_08_CS: Stop transaction options - EnergyTransfer stopped - StopAuthorized

Test case name	Stop transaction options - EnergyTransfer stopped - StopAuthorized
Test case Id	TC_E_08_CS
Use case Id(s)	E06(S6)
Requirement(s)	E06.FR.07, E06.FR.15
System under test	Charging Station
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the stop options.
Purpose	To verify if the Charging Station stops a transaction when the energy transfer stopped normally and it has been configured to do so.
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station does NOT have the following configuration; The mutability of TxStopPoint is <i>ReadOnly</i> AND (the value <i>EnergyTransfer</i> is NOT set OR (<i>Authorized</i> OR <i>PowerPathClosed</i>) is set). - If the mutability of TxStopPoint is <i>ReadWrite</i>, then the value <i>EnergyTransfer</i> must be supported.

Before (Preparations)
Configuration State: TxStopPoint contains <i>EnergyTransfer</i>
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
1. State is <i>StopAuthorized</i>	

Tool validations
* Step 1: N/a
Post scenario validations: N/a

TC_E_22_CS: Stop transaction options - EnergyTransfer stopped - SuspendedEV

Test case name	Stop transaction options - EnergyTransfer stopped - SuspendedEV
Test case Id	TC_E_22_CS
Use case Id(s)	E06(S6)
Requirement(s)	E06.FR.07, E06.FR.15
System under test	Charging Station
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the stop options.
Purpose	To verify if the Charging Station stops a transaction when the energy transfer stopped by the EV and the Charging Station has been configured to do so.
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station does NOT have the following configuration; The mutability of TxStopPoint is <i>ReadOnly</i> AND the value <i>EnergyTransfer</i> is NOT set. - If the mutability of TxStopPoint is <i>ReadWrite</i>, then the value <i>EnergyTransfer</i> must be supported.

Before (Preparations)
Configuration State: TxStopPoint contains <i>EnergyTransfer</i>
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action:</u> <i>The EV suspends the energy transfer.</i>	
1. The Charging Station sends a TransactionEventRequest	2. The Test System responds with a TransactionEventResponse

Tool validations
* Step 1: Message: TransactionEventRequest - triggerReason must be <i>ChargingStateChanged</i> - transactionInfo.chargingState must be <i>EVConnected</i> AND - transactionInfo.stoppedReason must be <i>StoppedByEV</i> AND - eventType must be <i>Ended</i>
Post scenario validations: N/a

TC_E_14_CS: Stop transaction options - EVDisconnected - Charging Station side

Test case name	Stop transaction options - EVDisconnected - Charging Station side
Test case Id	TC_E_14_CS
Use case Id(s)	E06(S2)
Requirement(s)	E06.FR.02, E06.FR.15
System under test	Charging Station
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the stop options.
Purpose	To verify if the Charging Station stops a transaction when the EV and EVSE are disconnected at the Charging Station side and it has been configured to do so.
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station does NOT have the following configuration; The mutability of TxStopPoint is <i>ReadOnly</i> AND (the value <i>EVConnected</i> is NOT set OR (<i>EnergyTransfer</i> OR <i>PowerPathClosed</i> OR <i>DataSigned</i> OR <i>Authorized</i> is set)). - If the mutability of TxStopPoint is <i>ReadWrite</i>, then the value <i>EVConnected</i> must be supported.

Before (Preparations)
Configuration State: TxStopPoint contains <i>EVConnected</i>
Memory State: N/a
Reusable State(s): State is <i>EVConnectedPostSession</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action</u> : Disconnect the EV and EVSE.	
1. The Charging Station notifies the CSMS about the status change of the connector.	2. The Test System responds accordingly.
3. The Charging Station sends a TransactionEventRequest	4. The Test System responds with a TransactionEventResponse

Tool validations
<p>* Step 1:</p> <p>Message: StatusNotificationRequest</p> <ul style="list-style-type: none"> - connectorStatus must be <i>Available</i> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none"> - eventData[0].trigger must be <i>Delta</i> - eventData[0].actualValue must be <i>Available</i> - eventData[0].component.name must be <i>Connector</i> - eventData[0].variable.name must be <i>AvailabilityState</i> <p>* Step 3:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - triggerReason must be <i>EVCommunicationLost</i> - transactionInfo.chargingState must be <i>Idle</i> <p>- If the Test System is configured to stop transactions using a <i>RequestStopTransactionRequest</i> message then transactionInfo.stoppedReason must be <i>Remote</i> or <i>EVDisconnected</i>.</p> <p>Else transactionInfo.stoppedReason must be <i>Local</i>, <i>EVDisconnected</i> or be omitted.</p> <ul style="list-style-type: none"> - eventType must be <i>Ended</i>
Post scenario validations: N/a

TC_E_20_CS: Stop transaction options - EVDisconnected - EV side (able to charge IEC 61851-1 EV)

Test case name	Stop transaction options - EVDisconnected - EV side (able to charge IEC 61851-1 EV)
Test case Id	TC_E_20_CS
Use case Id(s)	E06(S2), E10
Requirement(s)	E06.FR.02, E06.FR.15
System under test	Charging Station
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the stop options.
Purpose	To verify if the Charging Station stops a transaction when the EV and EVSE are disconnected at the EV side and it has been configured to do so.
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station does NOT have the following configuration; The mutability of TxStopPoint is <i>ReadOnly</i> AND (the value <i>EVConnected</i> is NOT set OR (<i>EnergyTransfer</i> OR <i>PowerPathClosed</i> is set)). - If the mutability of TxStopPoint is <i>ReadWrite</i>, then the value <i>EVConnected</i> must be supported. - The Charging Station does NOT have following configuration combination; StopTxOnEVSideDisconnect mutability <i>ReadOnly</i> with value <i>true</i> AND TxStopPoint mutability is <i>ReadOnly</i> and contains <i>Authorized</i>. - The Charging Station is able to charge with a EV that uses IEC 61851-1.

Before (Preparations)
Configuration State: TxStopPoint contains <i>EVConnected</i> StopTxOnEVSideDisconnect is <i>false</i> (If mutability is <i>ReadWrite</i>)
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferSuspended</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action</u> : Disconnect the EV and EVSE on EV side (EVSE loses connection with EV).	
1. The Charging Station sends a TransactionEventRequest	2. The Test System responds with a TransactionEventResponse

Tool validations
* Step 1: Message: TransactionEventRequest - triggerReason must be <i>EVCommunicationLost</i> - transactionInfo.chargingState must be <i>Idle</i> - transactionInfo.stoppedReason must be <i>EVDisconnected</i> - eventType must be <i>Ended</i>
Post scenario validations: N/a

TC_E_54_CS: Stop transaction options - EVDisconnected - EV side (not able to charge IEC 61851-1 EV)

Test case name	Stop transaction options - EVDisconnected - EV side (not able to charge IEC 61851-1 EV)
Test case Id	TC_E_54_CS
Use case Id(s)	E06(S2), E10
Requirement(s)	E06.FR.02, E06.FR.15
System under test	Charging Station
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the stop options.
Purpose	To verify if the Charging Station stops a transaction when the EV and EVSE are disconnected at the EV side and it has been configured to do so.
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station does NOT have the following configuration; The mutability of TxStopPoint is <i>ReadOnly</i> AND (the value <i>EVConnected</i> is NOT set OR (<i>EnergyTransfer</i> OR <i>PowerPathClosed</i> is set)). - If the mutability of TxStopPoint is <i>ReadWrite</i>, then the value <i>EVConnected</i> must be supported. - The Charging Station does NOT have following configuration combination; StopTxOnEVSideDisconnect mutability <i>ReadOnly</i> with value <i>true</i> AND TxStopPoint mutability is <i>ReadOnly</i> and contains <i>Authorized</i>. - The Charging Station supports high level communication.

Before (Preparations)
Configuration State: TxStopPoint contains <i>EVConnected</i> StopTxOnEVSideDisconnect is <i>false</i> (If mutability is <i>ReadWrite</i>)
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferSuspended</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action:</u> Disconnect the EV and EVSE on EV side (EVSE loses connection with EV).	
1. The Charging Station sends a TransactionEventRequest	2. The Test System responds with a TransactionEventResponse

Tool validations
* Step 1: Message: TransactionEventRequest - triggerReason must be <i>EVCommunicationLost</i> - transactionInfo.chargingState must be <i>Idle</i> - transactionInfo.stoppedReason must be <i>StoppedByEV</i> or <i>EVDisconnected</i> - eventType must be <i>Ended</i>
Post scenario validations: N/a

TC_E_15_CS: Stop transaction options - StopAuthorized - Local

Test case name	Stop transaction options - StopAuthorized - Local
Test case Id	TC_E_15_CS
Use case Id(s)	E06(S3)
Requirement(s)	E06.FR.03, E06.FR.15
System under test	Charging Station
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the stop options.
Purpose	To verify if the Charging Station stops a transaction when the EV driver locally stops the transaction and it has been configured to do so.
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station supports at least one authorization method described at the following Use cases; C01, C02, C04. - The Charging Station does NOT have the following configuration; The mutability of TxStopPoint is <i>ReadOnly</i> AND the value <i>Authorized</i> is NOT set OR <i>PowerPathClosed</i> is set. - If the mutability of TxStopPoint is <i>ReadWrite</i>, then the value <i>Authorized</i> must be supported.

Before (Preparations)
Configuration State: TxStopPoint contains <i>Authorized</i>
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferStarted</i>

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 Test System responds with a TransactionEventResponse

Tool validations
* Step 1: Message: TransactionEventRequest - triggerReason must be <i>StopAuthorized</i> - transactionInfo.stoppedReason must be <i>Local</i> - eventType must be <i>Ended</i>
Post scenario validations: N/a

TC_E_21_CS: Stop transaction options - StopAuthorized - Remote

Test case name	Stop transaction options - StopAuthorized - Remote
Test case Id	TC_E_21_CS
Use case Id(s)	E06(S3) AND F03
Requirement(s)	E06.FR.03, E06.FR.15 AND F03.FR.09
System under test	Charging Station
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the stop options.
Purpose	To verify if the Charging Station stops a transaction when it receives a RequestStopTransactionRequest and it has been configured to do so.
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station does NOT have the following configuration; The mutability of TxStopPoint is <i>ReadOnly</i> AND the value <i>Authorized</i> is NOT set OR <i>PowerPathClosed</i> is set. - If the mutability of TxStopPoint is <i>ReadWrite</i>, then the value <i>Authorized</i> 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 Test System sends a RequestStopTransactionRequest with transactionId <transactionId provided by the Charging Station in <i>TransactionEventRequest</i> >
3. The Charging Station sends a TransactionEventRequest	4. The Test System 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

TC_E_19_CS: Stop transaction options - ParkingBayUnoccupied

Test case name	Stop transaction options - ParkingBayUnoccupied
Test case Id	TC_E_19_CS
Use case Id(s)	E06(S1)
Requirement(s)	E06.FR.01, E06.FR.15
System under test	Charging Station
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the stop options.
Purpose	To verify if the Charging Station stops a transaction when the EV left the parking bay and it has been configured to do so.
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station does NOT have the following configuration; The mutability of TxStopPoint is <i>ReadOnly</i> AND (the value <i>ParkingBayOccupancy</i> is NOT set OR (<i>EnergyTransfer</i> OR <i>PowerPathClosed</i> OR <i>DataSigned</i> OR <i>Authorized</i> OR <i>EVConnected</i> is set)). - If the mutability of TxStopPoint is <i>ReadWrite</i>, then the value <i>ParkingBayOccupancy</i> must be supported.

Before (Preparations)
Configuration State: TxStopPoint contains <i>ParkingBayOccupancy</i>
Memory State: N/a
Reusable State(s): State is <i>EVDDisconnected</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action</u> : Drive EV out of parking bay.	
1. The Charging Station sends a TransactionEventRequest	2. The Test System responds with a TransactionEventResponse

Tool validations
* Step 1: Message: TransactionEventRequest - triggerReason must be <i>EVDeparted</i> - If the Test System is configured to stop transactions using a <i>RequestStopTransactionRequest</i> message then transactionInfo.stoppedReason must be <i>Remote</i> Else transactionInfo.stoppedReason must be <i>Local</i> - eventType must be <i>Ended</i>
Post scenario validations: N/a

TC_E_24_CS: Disconnect cable on EV-side - Deauthorize transaction - UnlockOnEVSideDisconnect is true

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 <i>true</i> UnlockOnEVSideDisconnect is <i>true</i>
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferSuspended</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action:</u> Disconnect the EV and EVSE on EV side (EVSE loses connection with EV).	
1. The Charging Station sends a TransactionEventRequest	2. The Test System responds with a TransactionEventResponse
<u>Manual Action:</u> 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 Test System responds accordingly.

Tool validations
<p>* Step 1:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - triggerReason must be <i>EVCommunicationLost</i> - transactionInfo.chargingState must be <i>Idle</i> <p>* Step 3:</p> <p>Message: StatusNotificationRequest</p> <ul style="list-style-type: none"> - connectorStatus <i>Available</i> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none"> - eventData[0].trigger <i>Delta</i> - eventData[0].actualValue <i>"Available"</i> - eventData[0].component.name <i>"Connector"</i> - eventData[0].variable.name <i>"AvailabilityState"</i> <p>Post scenario validations:</p> <p>N/a</p>

TC_E_25_CS: Disconnect cable on EV-side - Deauthorize transaction - UnlockOnEVSideDisconnect is false

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 <i>true</i> UnlockOnEVSideDisconnect is <i>false</i>
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferSuspended</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action</u> : Disconnect the EV and EVSE on EV side (EVSE loses connection with EV).	
1. The Charging Station sends a TransactionEventRequest	2. The Test System responds with a TransactionEventResponse
<u>Manual Action</u> : Present the IdToken that was used to start the transaction.	
<u>Note(s)</u> : - This manual action needs to be executed when the Charging Station has a detachable cable on the Charging Station side.	
<u>Manual Action</u> : Disconnect the EV and EVSE on Charging Station side.	
<u>Note(s)</u> : - 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 Test System responds accordingly.

Tool validations
* Step 1: Message: TransactionEventRequest - triggerReason must be <i>EVCommunicationLost</i> - transactionInfo.chargingState must be <i>Idle</i> * Step 3: Message: StatusNotificationRequest - connectorStatus <i>Available</i> Message: NotifyEventRequest - eventData[0].trigger <i>Delta</i> - eventData[0].actualValue <i>"Available"</i> - eventData[0].component.name <i>"Connector"</i> - eventData[0].variable.name <i>"AvailabilityState"</i>

Tool validations
Post scenario validations: N/a

TC_E_26_CS: Disconnect cable on EV-side - Suspend transaction

Test case name	Disconnect cable on EV-side - Suspend transaction
Test case Id	TC_E_26_CS
Use case Id(s)	E10
Requirement(s)	E10.FR.01, E10.FR.03
System under test	Charging Station
Description	The Charging Station can behave in several different ways when the cable is disconnected at the EV side, based on its configuration. This test case tests one of the possible configuration settings.
Purpose	To verify if the Charging Station suspends the transaction when the EV and EVSE are disconnected at the EV side and it has been configured to do so AND is able restart the energy transfer after reconnecting the EV and EVSE.
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station does NOT have the following configuration; The mutability of TxStopPoint is <i>ReadOnly</i> AND (the value <i>Authorized</i> OR <i>ParkingBayOccupancy</i> is NOT set OR (<i>EnergyTransfer</i> OR <i>PowerPathClosed</i> OR <i>DataSigned</i> OR <i>EVConnected</i> is set)). - If the mutability of TxStopPoint is <i>ReadWrite</i>, then the value <i>Authorized</i> OR <i>ParkingBayOccupancy</i> must be supported.

Before (Preparations)

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 (Test scenario)

Charging Station	CSMS
<u>Manual Action:</u> Disconnect the EV and EVSE on EV side (EVSE loses connection with EV).	
1. The Charging Station sends a TransactionEventRequest	2. The Test System responds with a TransactionEventResponse
3. The Charging Station notifies the CSMS about the current state of the connector.	4. The Test System responds accordingly.
<u>Note(s):</u> - This step needs to be executed when the Charging Station has a permanently attached cable on the Charging Station side.	
<u>Manual Action:</u> Reconnect the EV and EVSE on EV side.	
<u>Note(s):</u> - If the Charging Station has a permanently attached cable on the Charging Station side, then this step needs to be executed before the configured <i>EVConnectionTimeout</i> expires.	
5. The Charging Station sends a TransactionEventRequest	6. The Test System responds with a TransactionEventResponse
7. The Charging Station sends a TransactionEventRequest	8. The Test System responds with a TransactionEventResponse

Tool validations
<div>* Step 1: Message: TransactionEventRequest - triggerReason must be <i>EVCommunicationLost</i> - transactionInfo.chargingState must be <i>Idle</i> - eventType must be <i>Updated</i></div> <div>* Step 3: Message: StatusNotificationRequest - connectorStatus <i>Available</i> Message: NotifyEventRequest - eventData[0].trigger <i>Delta</i> - eventData[0].actualValue <i>"Available"</i> - eventData[0].component.name <i>"Connector"</i> - eventData[0].variable.name <i>"AvailabilityState"</i></div> <div>* Step 5: Message: TransactionEventRequest - triggerReason must be <i>CablePluggedIn</i> - transactionInfo.chargingState must be <i>EVConnected</i> - eventType must be <i>Updated</i></div> <div>* Step 7: Message: TransactionEventRequest - triggerReason must be <i>ChargingStateChanged</i> - transactionInfo.chargingState must be <i>Charging</i> - eventType must be <i>Updated</i></div>
<div>Post scenario validations: N/a</div>

TC_E_27_CS: Disconnect cable on EV-side - Suspend transaction - Fixed cable connection timeout

Test case name	Disconnect cable on EV-side - Suspend transaction - Fixed cable connection timeout
Test case Id	TC_E_27_CS
Use case Id(s)	E10
Requirement(s)	E10.FR.02, E10.FR.03
System under test	Charging Station
Description	The Charging Station can behave in several different ways when the cable is disconnected at the EV side, based on its configuration. This test case tests one of the possible configuration settings.
Purpose	To verify if the Charging Station suspends the transaction when the EV and EVSE are disconnected at the EV side and it has been configured to do so AND deauthorizes the transaction after the configured connection timeout expires.
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station does NOT have the following configuration; The mutability of TxStopPoint is <i>ReadOnly</i> AND (the value <i>Authorized</i> OR <i>ParkingBayOccupancy</i> is NOT set OR (<i>EnergyTransfer</i> OR <i>PowerPathClosed</i> OR <i>DataSigned</i> OR <i>EVConnected</i> is set)). - If the mutability of TxStopPoint is <i>_ReadWrite</i>, then the value <i>Authorized</i> OR <i>ParkingBayOccupancy</i> must be supported. - The Charging Station has a permanently attached cable at the Charging Station side. - StopTxOnEVSideDisconnect can be set to <i>false</i>.

Before (Preparations)
Configuration State: TxStopPoint contains <i>Authorized</i> (If supported) TxStopPoint contains <i>ParkingBayOccupancy</i> (If supported) UnlockOnEVSideDisconnect is <i>false</i> StopTxOnEVSideDisconnect is <i>false</i>
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferSuspended</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action:</u> <i>Disconnect the EV and EVSE on EV side (EVSE loses connection with EV).</i>	
1. The Charging Station sends a TransactionEventRequest	2. The Test System responds with a TransactionEventResponse
3. The Charging Station notifies the CSMS about the current state of the connector.	4. The Test System responds accordingly.
5. The Charging Station sends a TransactionEventRequest	6. The Test System responds with a TransactionEventResponse

Tool validations
<p>* Step 1:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none">- triggerReason must be <i>EVCommunicationLost</i>- transactionInfo.chargingState must be <i>Idle</i>- eventType must be <i>Updated</i> <p>* Step 3:</p> <p>Message: StatusNotificationRequest</p> <ul style="list-style-type: none">- connectorStatus <i>Available</i> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none">- eventData[0].trigger <i>Delta</i>- eventData[0].actualValue <i>"Available"</i>- eventData[0].component.name <i>"Connector"</i>- eventData[0].variable.name <i>"AvailabilityState"</i> <p>* Step 5:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none">- triggerReason must be <i>EVConnectTimeout</i> <p>If <Configured TxCtrlr.TxStopPoint> contains <i>Authorized</i> then</p> <p>eventType must be <i>Ended</i></p> <p>transactionInfo.stoppedReason must be <i>Timeout</i></p> <p>else if <Configured TxCtrlr.TxStopPoint> contains <i>ParkingBayOccupancy</i> then</p> <p>eventType must be <i>Updated</i></p>
<p>Post scenario validations:</p> <p>N/a</p>

TC_E_28_CS: Check Transaction status - TransactionId unknown

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 Test System sends a GetTransactionStatusRequest with transactionId <i><Randomly generated transactionId></i>

Tool validations
* Step 2: Message: GetTransactionStatusResponse - ongoingIndicator must be <i>false</i> - messagesInQueue must be <i>false</i>
Post scenario validations: N/a

TC_E_29_CS: Check Transaction status - Transaction with id ongoing - with message in queue

Test case name	Check Transaction status - Transaction with id ongoing - with message in queue
Test case Id	TC_E_29_CS
Use case Id(s)	E14
Requirement(s)	E14.FR.02,E14.FR.04
System under test	Charging Station
Description	The CSMS is able to request the status of a transaction and to find out whether there are queued transaction-related messages, using the GetTransactionStatusRequest message.
Purpose	To verify if the Charging Station is able to correctly respond to a GetTransactionStatusRequest with a transactionId, while there is a message queued belonging to an ongoing transaction with the requested id.
Prerequisite(s)	N/a

Before (Preparations)
<p>Configuration State:</p> <p>SampledDataTxUpdatedMeasurands is <Configured transaction_updated_metervalues_measurands></p> <p>SampledDataTxUpdatedInterval is <Configured transaction_updated_metervalues_interval></p> <p>OfflineThreshold is <Configured RetryBackOffWaitMinimum_duration> + 60.0</p> <p>RetryBackOffWaitMinimum is <Configured RetryBackOffWaitMinimum_duration></p> <p>RetryBackOffRandomRange is 0</p> <p><u>Note:</u> <Configured RetryBackOffWaitMinimum_duration> must be greater than <Configured Transaction MeterValues interval></p> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>EnergyTransferStarted</i></p>

Main (Test scenario)	
Charging Station	CSMS
	1. The Test System closes the WebSocket connection AND does not accept a reconnect.
	2. The Test System 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	<p>3. The Test System sends a GetTransactionStatusRequest with transactionId <Generated transactionId from Before></p> <p><u>Note:</u> This step will be executed the moment the WebSocket connection is restored.</p>
5. The Charging Stations sends a TransactionEventRequest	6. The Test System responds with a TransactionEventResponse
<p><u>Note(s):</u> - The Charging Station will empty its Transaction message queue. This will contain one or more TransactionEventRequest messages</p>	

Tool validations
<div>* Step 4: Message: GetTransactionStatusResponse<ul style="list-style-type: none">- ongoingIndicator must be <i>true</i>- messagesInQueue must be <i>true</i></div> <div>* Step 5: Message: TransactionEventRequest<ul style="list-style-type: none">- eventType must be <i>Updated</i>- meterValues must be present.- offline must be <i>true</i></div>
<div>Post scenario validations: N/a</div>

TC_E_30_CS: Check Transaction status - Transaction with id ongoing - without message in queue

Test case name	Check Transaction status - Transaction with id ongoing - without message in queue
Test case Id	TC_E_30_CS
Use case Id(s)	E14
Requirement(s)	E14.FR.02,E14.FR.05
System under test	Charging Station
Description	The CSMS is able to request the status of a transaction and to find out whether there are queued transaction-related messages, using the GetTransactionStatusRequest message.
Purpose	To verify if the Charging Station is able to correctly respond to a GetTransactionStatusRequest with a transactionId, while there is NO message queued belonging to an ongoing transaction with the requested id.
Prerequisite(s)	N/a

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a GetTransactionStatusResponse	1. The Test System sends a GetTransactionStatusRequest with transactionId <Generated transactionId from Before>

Tool validations
* Step 2: Message: GetTransactionStatusResponse - ongoingIndicator must be <i>true</i> - messagesInQueue must be <i>false</i>
Post scenario validations: N/a

TC_E_31_CS: Check Transaction status - Transaction with id ended - with message in queue

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)	<p>The following combination of conditions are NOT true:</p> <ul style="list-style-type: none"> - No local authorization methods are supported AND - TxStopPoint mutability is <i>false</i> and only contains Authorized AND - TxCtrlr.StopTxOnEVSideDisconnect mutability is <i>false</i> and value is <i>false</i> <p><i>Note: If conditions 2 and 3 are true, but condition 1 is false, then please configure Test System configuration <scenario> as local.</i></p>

Before (Preparations)
<p>Configuration State:</p> <p>SampledDataTxUpdatedMeasurands is <Configured transaction_updated_metervalues_measurands></p> <p>SampledDataTxUpdatedInterval is <Configured transaction_updated_metervalues_interval></p> <p>OfflineThreshold is <Configured RetryBackOffWaitMinimum_duration> + <Configured Transaction Duration> + 60.0</p> <p>RetryBackOffWaitMinimum is <Configured RetryBackOffWaitMinimum_duration> + <Configured Transaction Duration></p> <p>RetryBackOffRandomRange is 0</p> <p>UnlockOnEVSideDisconnect is <i>true</i> (If ReadWrite)</p> <p><u>Note:</u> <Configured Transaction Duration> should be long enough to execute manual tasks</p> <p>Memory State: N/a</p> <p>Reusable State(s): State is <i>EnergyTransferStarted</i></p>

Main (Test scenario)	
Charging Station	CSMS
	<i>The Test System closes the WebSocket connection AND does not accept a reconnect.</i>
<u>Manual Action:</u> Present the same idToken as used to start the transaction. <u>Notes(s):</u> Only if configured scenario is local	
<u>Manual Action:</u> Stop the energy transfer via the EV. <u>Notes(s):</u> Only if configured scenario is remote	
<u>Manual Action:</u> Disconnect the EV and EVSE.	
<u>Manual Action:</u> Drive EV out of parking bay. (Only needed if TxStopPoint is ParkingBayOccupancy)	
<u>Notes(s):</u> The tool will wait for <Configured Transaction Duration> seconds	
	<i>The Test System accepts reconnection attempt from the Charging Station.</i>
2. The Charging Station responds with a GetTransactionStatusResponse	1. The Test System sends a GetTransactionStatusRequest with transactionId <Generated transactionId from Before> <u>Note:</u> <i>This step will be executed the moment the WebSocket connection is restored.</i>

Main (Test scenario)	
3. The Charging Stations sends a <code>TransactionEventRequest</code> <u>Note(s):</u> - The Charging Station will empty its <i>Transaction message queue</i> . This will contain all <i>TransactionEventRequest</i> messages from the <i>Transaction</i> .	4. The Test System responds with a <code>TransactionEventResponse</code>

Tool validations
<p>* Step 2:</p> <p>Message: <code>GetTransactionStatusResponse</code></p> <ul style="list-style-type: none"> - <code>ongoingIndicator</code> must be <i>false</i> - <code>messagesInQueue</code> must be <i>true</i> <p>* Step 3:</p> <p>Message: <code>TransactionEventRequest</code></p> <p>The tool validations from the reusable states need to be used to verify whether all required <i>TransactionEventRequests</i> have been received.</p> <p>From <i>StopAuthorized</i> through <i>ParkingBayUnoccupied</i> (in case of scenario <i>Local</i>).</p> <p>And from <i>EnergyTransferSuspended</i> through <i>ParkingBayUnoccupied</i> (in case of scenario <i>Remote</i>).</p>
Post scenario validations: N/a

TC_E_32_CS: Check Transaction status - Transaction with id ended - without message in queue

Test case name	Check Transaction status - Transaction with id ended - without message in queue
Test case Id	TC_E_32_CS
Use case Id(s)	E14
Requirement(s)	E14.FR.03,E14.FR.05
System under test	Charging Station
Description	The CSMS is able to request the status of a transaction and to find out whether there are queued transaction-related messages, using the GetTransactionStatusRequest message.
Purpose	To verify if the Charging Station is able to correctly respond to a GetTransactionStatusRequest with a transactionId, while there is NO message queued belonging to an ended transaction with the requested id.
Prerequisite(s)	N/a

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferStarted</i> , <i>ParkingBayUnoccupied</i>

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a GetTransactionStatusResponse	1. The Test System sends a GetTransactionStatusRequest with transactionId <Generated transactionId from Before>

Tool validations
* Step 2: Message: GetTransactionStatusResponse - ongoingIndicator must be <i>false</i> - messagesInQueue must be <i>false</i>
Post scenario validations: N/a

TC_E_33_CS: Check Transaction status - Without transactionId - with message in queue

Test case name	Check Transaction status - Without transactionId - with message in queue
Test case Id	TC_E_33_CS
Use case Id(s)	E14
Requirement(s)	E14.FR.06,E14.FR.07
System under test	Charging Station
Description	The CSMS is able to request the status of a transaction and to find out whether there are queued transaction-related messages, using the GetTransactionStatusRequest message.
Purpose	To verify if the Charging Station is able to correctly respond to a GetTransactionStatusRequest without a transactionId, while there is a message queued.
Prerequisite(s)	N/a

Before (Preparations)
<p>Configuration State:</p> <p>SampledDataTxUpdatedMeasurands is <Configured transaction_updated_metervalues_measurands></p> <p>SampledDataTxUpdatedInterval is <Configured transaction_updated_metervalues_interval></p> <p>OfflineThreshold is <Configured RetryBackOffWaitMinimum_duration> + 60.0</p> <p>RetryBackOffWaitMinimum is <Configured RetryBackOffWaitMinimum_duration></p> <p>RetryBackOffRandomRange is 0</p> <p><u>Note:</u></p> <p><Configured RetryBackOffWaitMinimum_duration> must be greater than <Configured Transaction MeterValues interval></p> <p>Memory State:</p> <p>N/a</p> <p>Reusable State(s):</p> <p>State is <i>EnergyTransferStarted</i></p>

Main (Test scenario)	
Charging Station	CSMS
	1. The Test System closes the WebSocket connection AND does not accept a reconnect.
	2. The Test System 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	<p>3. The Test System sends a GetTransactionStatusRequest with transactionId omitted</p> <p><u>Note:</u></p> <p>This step will be executed the moment the WebSocket connection is restored.</p>
5. The Charging Stations sends a TransactionEventRequest	6. The Test System responds with a TransactionEventResponse
<p><u>Note(s):</u></p> <p>- The Charging Station will empty its Transaction message queue. This will contain one or more TransactionEventRequest messages</p>	

Tool validations
<p>* Step 4:</p> <p>Message: GetTransactionStatusResponse</p> <ul style="list-style-type: none">- ongoingIndicator must be omitted- messagesInQueue must be <i>true</i> <p>* Step 5:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none">- eventType must be <i>Updated</i>- meterValues must be present.- offline must be <i>true</i>
<p>Post scenario validations:</p> <p>N/a</p>

TC_E_34_CS: Check Transaction status - Without transactionId - without message in queue

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 Test System sends a GetTransactionStatusRequest with transactionId omitted

Tool validations
* Step 2: Message: GetTransactionStatusResponse - ongoingIndicator must be omitted - messagesInQueue must be <i>false</i>
Post scenario validations: N/a

TC_E_40_CS: Offline Behaviour - Connection loss during transaction

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 Test System closes the WebSocket connection AND does not accept a reconnect.
	2. The Test System 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	4. The Test System responds with a TransactionEventResponse
Note(s): - The Charging Station will empty its Transaction message queue. This will contain one or more TransactionEventRequest messages	

Tool validations

* Step 3:

All messages: **TransactionEventRequest**

- **eventType** must be *Updated*
- **meterValues** must be present.
- **offline** must be true

Post scenario validations:

N/a

TC_E_41_CS: Retry sending transaction message when failed - Max retry count reached

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
<u>Note(s)</u> : Step 1, 2, 3, & 4 are optional	
1. The Charging Stations sends a TransactionEventRequest with: - triggerReason <i>SignedDataReceived</i>	2. The Test System responds with a TransactionEventResponse
3. The Charging Stations sends a TransactionEventRequest with: - triggerReason <i>ChargingStateChanged</i> - chargingState <i>SuspendedEVSE</i>	4. The Test System responds with a TransactionEventResponse
<u>Note(s)</u> : 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 Test System 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

TC_E_50_CS: Retry sending transaction message when failed - Max retry count reached - CallError

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 <i>Authorized</i> State is <i>EVConnectedPreSession</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Note(s)</u> : Step 1, 2, 3, & 4 are optional	
1. The Charging Stations sends a TransactionEventRequest with: - triggerReason <i>SignedDataReceived</i>	2. The Test System responds with a TransactionEventResponse
3. The Charging Stations sends a TransactionEventRequest with: - triggerReason <i>ChargingStateChanged</i> - chargingState <i>SuspendedEVSE</i>	4. The Test System responds with a TransactionEventResponse
<u>Note(s)</u> : Step 5 is repeated for the configured number of times	
5. The Charging Stations sends a TransactionEventRequest	6. The Test System responds with a CallError with errorCode <i>InternalError</i>

Tool validations
* Step 1: - triggerReason <i>SignedDataReceived</i> * Step 3: - triggerReason <i>ChargingStateChanged</i> - chargingState <i>SuspendedEVSE</i> * 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 Test System 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

TC_E_42_CS: Retry sending transaction message when failed - Success before reaching the max retry count

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 <i>Authorized</i> State is <i>EVConnectedPreSession</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Note(s)</u> : Step 1, 2, 3, & 4 are optional	
1. The Charging Stations sends a TransactionEventRequest with: - triggerReason <i>SignedDataReceived</i>	2. The Test System responds with a TransactionEventResponse
3. The Charging Stations sends a TransactionEventRequest with: - triggerReason <i>ChargingStateChanged</i> - chargingState <i>SuspendedEVSE</i>	4. The Test System responds with a TransactionEventResponse
<u>Note(s)</u> : The tool will ignore the first request and only respond to the second request	
5. The Charging Stations sends a TransactionEventRequest	6. The Test System responds with a TransactionEventResponse

Tool validations
* Step 1: - triggerReason <i>SignedDataReceived</i> * Step 3: - triggerReason <i>ChargingStateChanged</i> - chargingState <i>SuspendedEVSE</i> * 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) + <i>OcppCommCtrlr.MessageTimeout.Default</i> . - The Test System 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

TC_E_51_CS: Retry sending transaction message when failed - Success before reaching the max retry count - CallError

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 <i>Authorized</i> State is <i>EVConnectedPreSession</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Note(s)</u> : Step 1, 2, 3, & 4 are optional	
1. The Charging Stations sends a TransactionEventRequest with: - triggerReason <i>SignedDataReceived</i>	2. The Test System responds with a TransactionEventResponse
3. The Charging Stations sends a TransactionEventRequest with: - triggerReason <i>ChargingStateChanged</i> - chargingState <i>SuspendedEVSE</i>	4. The Test System responds with a TransactionEventResponse
<u>Note(s)</u> : The tool will send a CallError with <i>errorCode InternalError</i> to all requests except for the second request, there a TransactionEventResponse is send	
5. The Charging Stations sends a TransactionEventRequest	6. The Test System responds with a TransactionEventResponse

Tool validations
* Step 1: - triggerReason <i>SignedDataReceived</i> * Step 3: - triggerReason <i>ChargingStateChanged</i> - chargingState <i>SuspendedEVSE</i> * 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) + <i>OcppCommCtrlr.MessageTimeout.Default</i> . - The Test System 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

TC_E_43_CS: Offline Behaviour - Transaction during offline period

Test case name	Offline Behaviour - Transaction during offline period
Test case Id	TC_E_43_CS
Use case Id(s)	E12
Requirement(s)	E12.FR.01,E12.FR.02,E12.FR.06
System under test	Charging Station
Description	The Charging Station queues TransactionEvent messages to inform the CSMS that a transaction occurred while the Charging Station was Offline.
Purpose	To verify if the Charging Station is able to queue TransactionEvent messages while it was offline.
Prerequisite(s)	The Charging Station supports authorization methods other than NoAuthorization

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>TransactionEventsInQueueEnded</i>

Main (Test scenario)	
Charging Station	CSMS
1. The Charging Stations sends a TransactionEventRequest	2. The Test System responds with a TransactionEventResponse
<u>Note(s):</u> - The Charging Station will empty its Transaction message queue. This will contain one or more TransactionEventRequest messages	

Tool validations
<p>* Step 1:</p> <p>All messages: TransactionEventRequest</p> <p>- offline must be <i>true</i></p> <p>One of the messages: TransactionEventRequest</p> <p>- eventType <i>Started</i></p> <p>One of the messages: TransactionEventRequest</p> <p>- eventType <i>Ended</i></p>
Post scenario validations: N/a

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

TC_E_44_CS: Offline Behaviour - Stop transaction during offline period

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 <u>Note:</u> <Configured RetryBackOffWaitMinimum_duration should be long enough to execute manual tasks>
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
	1. The Test System closes the WebSocket connection AND does not accept a reconnect.
<u>Manual Action:</u> Present the same idToken as used to start the transaction.	
<u>Manual Action:</u> Disconnect the EV and EVSE.	
<u>Manual Action:</u> Drive EV out of parking bay. (Only needed if TxStopPoint is ParkingBayOccupancy)	
	2. The Test System accepts the reconnection attempt from the Charging Station.
3. The Charging Stations sends a TransactionEventRequest	4. The Test System responds with a TransactionEventResponse
<u>Note(s):</u> - The Charging Station will empty its Transaction message queue. This will contain one or more TransactionEventRequest messages	

Tool validations
* Step 3: All messages: TransactionEventRequest - offline must be <i>true</i> One of the messages: TransactionEventRequest - eventType <i>Ended</i>
Post scenario validations: N/a

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

TC_E_45_CS: Offline Behaviour - Stop transaction during offline period - Same GroupId

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)	<ul style="list-style-type: none"> - 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 Test System closes the WebSocket connection AND does not accept a reconnect.
<u>Manual Action:</u> Present <Configured valid idtoken fields2>.	
<u>Manual Action:</u> Disconnect the EV and EVSE.	
<u>Manual Action:</u> Drive EV out of parking bay. (Only needed if TxStopPoint is ParkingBayOccupancy)	
	2. The Test System accepts the reconnection attempt from the Charging Station.
3. The Charging Stations sends a TransactionEventRequest	4. The Test System responds with a TransactionEventResponse
<u>Note(s):</u> - The Charging Station will empty its Transaction message queue. This will contain one or more TransactionEventRequest messages	

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.

TC_E_46_CS: End of charging process 15118

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 <i>EVConnectedPreSession</i> State is <i>Authorized15118</i> State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Note:</u> The Charging Station receives a SessionStopReq(Terminate) message from the EV.	
1. The Charging Station sends a TransactionEventRequest	2. The Test System responds with a TransactionEventResponse

Tool validations
<p>* Step 1:</p> <p>Message: TransactionEventRequest</p> <p>If <Configured TxStopPoint> contains "Authorized" or "PowerPathClosed" or "EnergyTransfer":</p> <ul style="list-style-type: none"> - eventType is <i>Ended</i> - triggerReason is <i>StopAuthorized</i> - transactionInfo.stoppedReason is <i>StoppedByEV</i> - transactionInfo.chargingState is <i>EVConnected</i> <p>If <Configured TxStopPoint> does not contain "Authorized" or "PowerPathClosed" or "EnergyTransfer":</p> <ul style="list-style-type: none"> - eventType is <i>Updated</i> - triggerReason = <i>StopAuthorized</i> - transactionInfo.chargingState is <i>EVConnected</i>
Post scenario validations: N/a

TC_E_100_CS: Transactions with fixed cost, energy or time - CSMS specifies energy limit

Test case name	Transactions with fixed cost, energy or time - CSMS specifies energy limit
Test case Id	TC_E_100_CS
Use case Id(s)	E16
Requirement(s)	E16.FR.02, E16.FR.05, E16.FR.06, E16.FR.08
System under test	Charging Station
Description	CSMS will limit the transaction to the specified energy limit.
Purpose	To verify whether the Charging Station uses the specified energy limit to limit the transaction.
Prerequisite(s)	TxCtrlr.SupportedLimits contains <i>maxEnergy</i>

Before (Preparations)
Configuration State: SampledDataTxUpdatedInterval is <i><Configured sampled Meter Values Updated interval></i>
Memory State: N/a
Reusable State(s): State is <i>Authorized</i> State is <i>EVConnectedPreSession</i>

Main (Test scenario)	
Charging Station	CSMS
1. The Charging Station sends a TransactionEventRequest	2. The Test System responds with a TransactionEventResponse with transactionLimit.maxEnergy 2000 transactionLimit.maxTime <i><omitted></i> transactionLimit.maxCost <i><omitted></i>
3. The Charging Station sends a TransactionEventRequest	4. The Test System responds with a TransactionEventResponse
5. The Charging Station sends a TransactionEventRequest	6. The Test System responds with a TransactionEventResponse with transactionLimit.maxEnergy 200 transactionLimit.maxTime <i><omitted></i> transactionLimit.maxCost <i><omitted></i>
7. The Charging Station sends a TransactionEventRequest	8. The Test System responds with a TransactionEventResponse
9. The Charging Station sends a TransactionEventRequest	10. The Test System responds with a TransactionEventResponse
<u>Note:</u> Steps 9 and 10 may repeat	
11. The Charging Station sends a TransactionEventRequest	12. The Test System responds with a TransactionEventResponse
<u>Note:</u> Step 11 should be triggered when 200 Wh has been charged	

Tool validations
<p>* Step 1:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - transactionInfo.transactionLimit must be <i><omitted></i> <p>* Step 3:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - eventType must be <i>Updated</i> - triggerReason must be <i>LimitSet</i> - transactionInfo.transactionLimit.maxEnergy must be <i>2000</i> - transactionInfo.transactionLimit.maxTime must be <i><omitted></i> - transactionInfo.transactionLimit.maxCost must be <i><omitted></i> <p>* Step 5:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - eventType must be <i>Updated</i> - transactionInfo.transactionLimit must be <i><omitted></i> <p>* Step 7:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - eventType must be <i>Updated</i> - triggerReason must be <i>LimitSet</i> - transactionInfo.transactionLimit.maxEnergy must be <i>200</i> - transactionInfo.transactionLimit.maxTime must be <i><omitted></i> - transactionInfo.transactionLimit.maxCost must be <i><omitted></i> <p>* Step 9:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - eventType must be <i>Updated</i> - transactionInfo.transactionLimit must be <i><omitted></i> <p>* Step 11:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - eventType must be <i>Updated</i> (or <i>Ended</i> if TxStopPoint contains "EnergyTransfer") - triggerReason must be <i>EnergyLimitReached</i> - transactionInfo.transactionLimit.maxEnergy must be <i>200</i> - transactionInfo.transactionLimit.maxTime must be <i><omitted></i> - transactionInfo.transactionLimit.maxCost must be <i><omitted></i> - transactionInfo.chargingState must be <i>SuspendedEVSE</i> (if TxStopPoint does not contain "EnergyTransfer")
<p>Post scenario validations:</p> <p>N/a</p>

TC_E_101_CS: Transactions with fixed cost, energy or time - CSMS calculates costs (through CostUpdatedRequest) and CS specifies limit

Test case name	Transactions with fixed cost, energy or time - CSMS calculates costs (through CostUpdatedRequest) and CS specifies limit
Test case Id	TC_E_101_CS
Use case Id(s)	E16
Requirement(s)	E16.FR.03, E16.FR.05, E16.FR.06, E16.FR.08, E16.FR.11
System under test	Charging Station
Description	CS will set a limit the transaction for the specified cost. CS will use running cost calculation provided by CSMS.
Purpose	To verify whether the Charging Station uses the running cost calculation provided by CSMS.
Prerequisite(s)	CS supports user setting a maximum cost limit. TxCtrlr.SupportedLimits contains <i>maxCost</i> .

Before (Preparations)
Configuration State: TariffCostCtrlr.Enabled[Tariff] is <i>false</i> TariffCostCtrlr.Enabled[Cost] is <i>false</i> SampledDataTxUpdatedInterval is <i><Configured sampled Meter Values Updated interval></i>
Memory State: N/a
Reusable State(s): State is <i>Authorized</i> State is <i>EVConnectedPreSession</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action:</u> Enter cost limit of 45.30 for transaction	
1. The Charging Station sends a TransactionEventRequest	2. The Test System responds with a TransactionEventResponse
4. The Charging Station responds with a CostUpdatedResponse	3. The Test System sends a CostUpdatedRequest with totalCost 44.30
5. The Charging Station sends a TransactionEventRequest	6. The Test System responds with a TransactionEventResponse
8. The Charging Station responds with a CostUpdatedResponse	7. The Test System sends a CostUpdatedRequest with totalCost 45.30
9. The Charging Station sends a TransactionEventRequest	10. The Test System responds with a TransactionEventResponse

Tool validations
<p>* Step 1:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - eventType must be <i>Updated</i> - triggerReason must be <i>LimitSet</i> - transactionInfo.transactionLimit.maxEnergy must be <i><omitted></i> - transactionInfo.transactionLimit.maxTime must be <i><omitted></i> - transactionInfo.transactionLimit.maxCost must be <i>45.30</i> <p>* Step 5:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - eventType must be <i>Updated</i> - triggerReason must not be <i>LimitSet</i> - transactionInfo.transactionLimit must be <i><omitted></i> <p>* Step 9:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - eventType must be <i>Updated</i> (or <i>Ended</i> if TxStopPoint contains "EnergyTransfer") - triggerReason must be <i>CostLimitReached</i> - transactionInfo.transactionLimit must be <i><omitted></i> - transactionInfo.chargingState must be <i>SuspendedEVSE</i> (or <i>EVConnected</i> if TxStopPoint contains "EnergyTransfer")
<p>Post scenario validations:</p> <p>N/a</p>

TC_E_102_CS: Transactions with fixed cost, energy or time - CSMS and CS both specify limits

Test case name	Transactions with fixed cost, energy or time - CSMS and CS both specify limits
Test case Id	TC_E_102_CS
Use case Id(s)	E16
Requirement(s)	E16.FR.02, E16.FR.03, E16.FR.04, E16.FR.05, E16.FR.06, E16.FR.08
System under test	Charging Station
Description	CSMS will limit the transaction to the specified energy and time limit. CS will add its own limits for energy.
Purpose	To verify whether the Charging Station uses the most limiting specified limits.
Prerequisite(s)	CS allows user to enter a maximum energy limit. TxCtrlr.SupportedLimits contains <i>maxEnergy</i> .

Before (Preparations)
Configuration State: SampledDataTxUpdatedInterval is <Configured sampled Meter Values Updated interval>
Memory State: N/a
Reusable State(s): State is <i>Authorized</i> State is <i>EVConnectedPreSession</i>

Main (Test scenario)	
Charging Station	CSMS
1. The Charging Station sends a TransactionEventRequest	2. The Test System responds with a TransactionEventResponse with transactionLimit.maxEnergy 20000
3. The Charging Station sends a TransactionEventRequest	4. The Test System responds with a TransactionEventResponse
5. The Charging Station sends a TransactionEventRequest	6. The Test System responds with a TransactionEventResponse
<u>Note:</u> Steps 5 and 6 may repeat during manual action	
<u>Manual Action:</u> Enter energy limit 10 kWh for transaction	
7. The Charging Station sends a TransactionEventRequest	8. The Test System responds with a TransactionEventResponse
9. The Charging Station sends a TransactionEventRequest	10. The Test System responds with a TransactionEventResponse
<u>Note:</u> Steps 9 and 10 may repeat during manual action	
<u>Manual Action:</u> Enter energy limit 40 kWh for transaction	
<u>Note:</u> CS should not allow this or at least not send a higher limit than 20000 as set by CSMS	
11. The Charging Station sends a TransactionEventRequest	12. The Test System responds with a TransactionEventResponse
13. The Charging Station sends a TransactionEventRequest	14. The Test System responds with a TransactionEventResponse with transactionLimit.maxEnergy 2000
15. The Charging Station sends a TransactionEventRequest	16. The Test System responds with a TransactionEventResponse

Main (Test scenario)	
17. The Charging Station sends a TransactionEventRequest	18. The Test System responds with a TransactionEventResponse
<u>Note</u> : Steps 17 and 18 may repeat	
19. The Charging Station sends a TransactionEventRequest	20. The Test System responds with a TransactionEventResponse
<u>Note</u> : Step 19 should be triggered when 2kWh has been charged	

Tool validations

* Step 1:

Message: **TransactionEventRequest**

- **transactionInfo.transactionLimit** must be *<omitted>*

* Step 3:

Message: **TransactionEventRequest**

- **eventType** must be *Updated*

- **triggerReason** must be *LimitSet*

- **transactionInfo.transactionLimit.maxEnergy** must be *20000*

- **transactionInfo.transactionLimit.maxTime** must be *3600*

- **transactionInfo.transactionLimit.maxCost** must be *<omitted>*

* Step 5:

Message: **TransactionEventRequest**

- **eventType** must be *Updated*

- **transactionInfo.transactionLimit** must be *<omitted>*

* Step 7:

Message: **TransactionEventRequest**

- **eventType** must be *Updated*

- **triggerReason** must be *LimitSet*

- **transactionInfo.transactionLimit.maxEnergy** must be *10000*

- **transactionInfo.transactionLimit.maxTime** must be *3600*

- **transactionInfo.transactionLimit.maxCost** must be *<omitted>*

* Step 11:

Message: **TransactionEventRequest**

- **eventType** must be *Updated*

IF **triggerReason** is *LimitSet*

THEN

- **transactionInfo.transactionLimit.maxEnergy** must be *20000*

- **transactionInfo.transactionLimit.maxTime** must be *3600*

- **transactionInfo.transactionLimit.maxCost** must be *<omitted>*

ENDIF

* Step 13:

Message: **TransactionEventRequest**

- **eventType** must be *Updated*

- **transactionInfo.transactionLimit** must be *<omitted>*

* Step 15:

Message: **TransactionEventRequest**

- **eventType** must be *Updated*

- **triggerReason** must be *LimitSet*

- **transactionInfo.transactionLimit.maxEnergy** must be *2000*

- **transactionInfo.transactionLimit.maxTime** must be *3600*

- **transactionInfo.transactionLimit.maxCost** must be *<omitted>*

* Step 19:

Message: **TransactionEventRequest**

- **eventType** must be *Updated* (or *Ended* if TxStopPoint contains "EnergyTransfer")

- **triggerReason** must be *EnergyLimitReached*

- **transactionInfo.transactionLimit** must be *<omitted>*

- **transactionInfo.chargingState** must be *SuspendedEVSE* (or *EVConnected* if TxStopPoint contains "EnergyTransfer")

Post scenario validations:

N/a

TC_E_103_CS: Transactions with fixed cost, energy or time - CS calculates costs and CSMS specifies limit

Test case name	Transactions with fixed cost, energy or time - CS calculates costs and CSMS specifies limit
Test case Id	TC_E_103_CS
Use case Id(s)	E16
Requirement(s)	E16.FR.02, E16.FR.05, E16.FR.06, E16.FR.08, E16.FR.11
System under test	Charging Station
Description	CSMS will set a limit the transaction for the specified cost. CS will use local cost calculation.
Purpose	To verify whether the Charging Station correctly uses local cost calculation.
Prerequisite(s)	CS supports local cost calculation TxCtrlr.SupportedLimits contains <i>maxCost</i>

Before (Preparations)

Configuration State:

TariffCostCtrlr.Enabled[Tariff] is *true*

TariffCostCtrlr.Enabled[Cost] is *true*

SampledDataTxUpdatedInterval is *<Configured sampled Meter Values Updated interval>*

Memory State:

N/a

Reusable State(s):

State is *EVConnectedPreSession*

Main (Test scenario)

Charging Station	CSMS
2. The Charging Station responds with a SetDefaultTariffResponse	1. The Test System sends a SetDefaultTariffRequest with evseld 0 tariff.tariffId <i>Tariff1</i> tariff.currency <i>EUR</i> tariff.chargingTime.taxRate[0].type <i>MyTax1</i> tariff.chargingTime.taxRate[0].tax 0 tariff.chargingTime.prices[0].priceMinute 60 tariff.idleTime.taxRate[0].type <i>MyTax2</i> tariff.idleTime.taxRate[0].tax 0 tariff.idleTime.prices[0].priceMinute 60
3. Execute Reusable State <i>EnergyTransferStarted</i>	
<u>Manual Action</u> : Enter cost limit 120.00 for transaction	
4. The Charging Station sends a TransactionEventRequest	5. The Test System responds with a TransactionEventResponse with transactionLimit.maxCost 120.00
6. The Charging Station sends a TransactionEventRequest	7. The Test System responds with a TransactionEventResponse
<u>Note</u> : Steps 6 and 7 may repeat	
8. The Charging Station sends a TransactionEventRequest	9. The Test System responds with a TransactionEventResponse
<u>Note</u> : Step 10 should trigger automatically after 120 seconds have been elapsed since start of transaction	

Tool validations
<p>* Step 4:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - eventType must be <i>Updated</i> - triggerReason must be <i>LimitSet</i> - transactionInfo.transactionLimit.maxEnergy must be <i><omitted></i> - transactionInfo.transactionLimit.maxTime must be <i><omitted></i> - transactionInfo.transactionLimit.maxCost must be <i>120.00</i> <p>* Step 6:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - eventType must be <i>Updated</i> - transactionInfo.transactionLimit must be <i><omitted></i>
<p>* Step 8:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - eventType must be <i>Updated</i> (or <i>Ended</i> if TxStopPoint contains "EnergyTransfer") - triggerReason must be <i>EnergyLimitReached</i> - transactionInfo.transactionLimit must be <i><omitted></i> - costDetails.totalCost.currency must be <i>EUR</i> - costDetails.totalCost.typeOfCost must be <i>NormalCost</i> - costDetails.totalCost.total.exclTax must be <i><not omitted></i> - costDetails.totalCost.total.inclTax must be <i><not omitted></i> - costDetails.totalCost.chargingTime.exclTax must be <i><not omitted></i> - costDetails.totalCost.chargingTime.inclTax must be <i><not omitted></i> - costDetails.totalCost.chargingTime.taxRates[0].type must be <i>MyTax1</i> - costDetails.totalCost.chargingTime.taxRates[0].rate must be <i>0</i> - costDetails.totalCost.idleTime.exclTax must be <i><not omitted></i> - costDetails.totalCost.idleTime.inclTax must be <i><not omitted></i> - costDetails.totalCost.idleTime.taxRates[0].type must be <i>MyTax2</i> - costDetails.totalCost.idleTime.taxRates[0].rate must be <i>0</i> - costDetails.totalUsage.chargingTime must be <i><not omitted></i> - costDetails.totalUsage.idleTime must be <i><not omitted></i> - transactionInfo.chargingState must be <i>SuspendedEVSE</i> (or <i>EVConnected</i> if TxStopPoint contains "EnergyTransfer")
<p>Post scenario validations:</p> <p>N/a</p>

TC_E_104_CS: Transactions with fixed cost, energy or time - CSMS calculates costs (through TransactionEventResponse) and specifies limit

Test case name	Transactions with fixed cost, energy or time - CSMS calculates costs (through TransactionEventResponse) and specifies limit
Test case Id	TC_E_104_CS
Use case Id(s)	E16
Requirement(s)	E16.FR.02, E16.FR.05, E16.FR.06, E16.FR.07, E16.FR.08, E16.FR.11
System under test	Charging Station
Description	CSMS will set a limit the transaction for the specified cost. CS will use running cost calculation provided by CSMS.
Purpose	To verify whether the Charging Station uses the running cost calculation provided by CSMS.
Prerequisite(s)	TxCtrlr.SupportedLimits contains <i>maxCost</i>

Before (Preparations)
Configuration State: TariffCostCtrlr.Enabled[Tariff] is <i>false</i> TariffCostCtrlr.Enabled[Cost] is <i>false</i> SampledDataTxUpdatedInterval is <i><Configured sampled Meter Values Updated interval></i>
Memory State: N/a
Reusable State(s): State is <i>Authorized</i>

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>EVConnectedPreSession</i> with: (At the first TransactionEventResponse) - transactionLimit.maxCost 99.13	
2. Execute Reusable State <i>EnergyTransferStarted</i> with: (At the first TransactionEventResponse) - transactionLimit.maxCost 99.13	
3. The Charging Station sends a TransactionEventRequest	4. The Test System responds with a TransactionEventResponse with totalCost 50.34
5. The Charging Station sends a TransactionEventRequest	6. The Test System responds with a TransactionEventResponse with totalCost 120.34
7. The Charging Station sends a TransactionEventRequest	8. The Test System responds with a TransactionEventResponse
<u>Note:</u> Step 7 should be triggered by the TransactionEventResponse of step 6	

Tool validations
<p>* Step 3:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - eventType must be <i>Updated</i> - triggerReason must be <i>LimitSet</i> - transactionInfo.transactionLimit.maxEnergy must be <i><omitted></i> - transactionInfo.transactionLimit.maxTime must be <i><omitted></i> - transactionInfo.transactionLimit.maxCost must be <i>99.13</i> <p>* Step 5:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - eventType must be <i>Updated</i> - transactionInfo.transactionLimit must be <i><omitted></i> <p>* Step 7:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - eventType must be <i>Updated</i> (or <i>Ended</i> if TxStopPoint contains "EnergyTransfer") - triggerReason must be <i>CostLimitReached</i> - transactionInfo.transactionLimit must be <i><omitted></i> - transactionInfo.chargingState must be <i>SuspendedEVSE</i> (or <i>_EVConnected</i> if TxStopPoint contains "EnergyTransfer")
<p>Post scenario validations:</p> <p>N/a</p>

TC_E_105_CS: Transactions with fixed cost, energy or time - CSMS specifies time limit

Test case name	Transactions with fixed cost, energy or time - CSMS specifies time limit
Test case Id	TC_E_105_CS
Use case Id(s)	E16
Requirement(s)	E16.FR.02, E16.FR.05, E16.FR.06, E16.FR.08, E16.FR.09
System under test	Charging Station
Description	CSMS will limit the transaction to the specified time limit.
Purpose	To verify whether the Charging Station uses the specified time limit to limit the transaction.
Prerequisite(s)	TxCtrlr.SupportedLimits contains <i>maxTime</i>

Before (Preparations)
Configuration State: SampledDataTxUpdatedInterval is <Configured sampled Meter Values Updated interval>
Memory State: N/a
Reusable State(s): State is <i>Authorized</i> State is <i>EVConnectedPreSession</i>

Main (Test scenario)	
Charging Station	CSMS
1. The Charging Station sends a TransactionEventRequest	2. The Test System responds with a TransactionEventResponse with transactionLimit.maxTime 3600
3. The Charging Station sends a TransactionEventRequest	4. The Test System responds with a TransactionEventResponse with
5. The Charging Station sends a TransactionEventRequest	6. The Test System responds with a TransactionEventResponse with transactionLimit.maxTime 300
7. The Charging Station sends a TransactionEventRequest	8. The Test System responds with a TransactionEventResponse
9. The Charging Station sends a TransactionEventRequest	10. The Test System responds with a TransactionEventResponse
<u>Note:</u> Steps 9 and 10 may repeat	
11. The Charging Station sends a TransactionEventRequest	12. The Test System responds with a TransactionEventResponse
<u>Note:</u> Step 11 should be triggered when 300 seconds have elapsed since start of transaction	

Tool validations
<p>* Step 1:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - transactionInfo.transactionLimit must be <i><omitted></i> <p>* Step 3:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - eventType must be <i>Updated</i> - triggerReason must be <i>LimitSet</i> - transactionInfo.transactionLimit.maxEnergy must be <i><omitted></i> - transactionInfo.transactionLimit.maxTime must be <i>3600</i> - transactionInfo.transactionLimit.maxCost must be <i><omitted></i> <p>* Step 5:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - eventType must be <i>Updated</i> - transactionInfo.transactionLimit must be <i><omitted></i> <p>* Step 7:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - eventType must be <i>Updated</i> - triggerReason must be <i>LimitSet</i> - transactionInfo.transactionLimit.maxEnergy must be <i><omitted></i> - transactionInfo.transactionLimit.maxTime must be <i>300</i> - transactionInfo.transactionLimit.maxCost must be <i><omitted></i> <p>* Step 9:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - eventType must be <i>Updated</i> - transactionInfo.transactionLimit must be <i><omitted></i> <p>* Step 11:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - eventType must be <i>Updated</i> (or <i>Ended</i> if TxStopPoint contains "EnergyTransfer") - triggerReason must be <i>EnergyLimitReached</i> - transactionInfo.transactionLimit must be <i><omitted></i> - transactionInfo.chargingState must be <i>SuspendedEVSE</i> (or <i>EVConnected</i> if TxStopPoint contains "EnergyTransfer")
<p>Post scenario validations:</p> <p>N/a</p>

TC_E_106_CS: Transactions with fixed cost, energy or time - CS specifies energy limit

Test case name	Transactions with fixed cost, energy or time - CS specifies energy limit
Test case Id	TC_E_106_CS
Use case Id(s)	E16
Requirement(s)	E16.FR.01, E16.FR.05, E16.FR.06
System under test	Charging Station
Description	The EV Driver is able to specify an energy limit.
Purpose	To verify whether the Charging Station uses the specified energy limit to limit the transaction.
Prerequisite(s)	- Charging Station supports manual entering of maximum energy amount. - TxCtrlr.SupportedLimits contains <i>maxEnergy</i>

Before (Preparations)
Configuration State: SampledDataTxUpdatedInterval is <Configured sampled Meter Values Updated interval>
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action:</u> Enter energy limit 20 kWh for transaction	
1. The Charging Station sends a TransactionEventRequest	2. The Test System responds with a TransactionEventResponse
3. The Charging Station sends a TransactionEventRequest	4. The Test System responds with a TransactionEventResponse
<u>Manual Action:</u> Change energy limit to 2 kWh	
<u>Note:</u> Steps 3 and 4 may repeat while entering new limit	
5. The Charging Station sends a TransactionEventRequest	6. The Test System responds with a TransactionEventResponse
7. The Charging Station sends a TransactionEventRequest	8. The Test System responds with a TransactionEventResponse
<u>Note:</u> Steps 7 and 8 may repeat until 2 kWh energy reached	
9. The Charging Station sends a TransactionEventRequest	10. The Test System responds with a TransactionEventResponse
<u>Note:</u> Step 9 should be triggered when 2kWh has been charged	

Tool validations
<p>* Step 1:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - eventType must be <i>Updated</i> - triggerReason must be <i>LimitSet</i> or <i>MeterValuePeriodic</i> - transactionInfo.transactionLimit.maxEnergy must be <i>20000</i> - transactionInfo.transactionLimit.maxTime must be <i><omitted></i> - transactionInfo.transactionLimit.maxCost must be <i><omitted></i> <p>* Step 3:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - eventType must be <i>Updated</i> - transactionInfo.transactionLimit must be <i><omitted></i> <p>* Step 5:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - eventType must be <i>Updated</i> - triggerReason must be <i>LimitSet</i> or <i>MeterValuePeriodic</i> - transactionInfo.transactionLimit.maxEnergy must be <i>2000</i> - transactionInfo.transactionLimit.maxTime must be <i><omitted></i> - transactionInfo.transactionLimit.maxCost must be <i><omitted></i> <p>* Step 9:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - eventType must be <i>Updated</i> (or <i>Ended</i> if TxStopPoint contains "EnergyTransfer") - triggerReason must be <i>EnergyLimitReached</i> - transactionInfo.transactionLimit.maxEnergy must be <i>2000</i> - transactionInfo.transactionLimit.maxTime must be <i><omitted></i> - transactionInfo.transactionLimit.maxCost must be <i><omitted></i> - transactionInfo.chargingState must be <i>SuspendedEVSE</i> (or <i>EVConnected</i> if TxStopPoint contains "EnergyTransfer")
<p>Post scenario validations:</p> <p>N/a</p>

TC_E_107_CS: Transactions with fixed cost, energy or time - CS specifies time limit

Test case name	Transactions with fixed cost, energy or time - CS specifies time limit
Test case Id	TC_E_107_CS
Use case Id(s)	E16
Requirement(s)	E16.FR.01, E16.FR.05, E16.FR.06
System under test	Charging Station
Description	The EV Driver is able to specify an time limit.
Purpose	To verify whether the Charging Station uses the specified time limit to limit the transaction.
Prerequisite(s)	- Charging Station supports manual entering of maximum time amount. - TxCtrlr.SupportedLimits contains <i>maxTime</i>

Before (Preparations)
Configuration State: SampledDataTxUpdatedInterval is <Configured sampled Meter Values Updated interval>
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action:</u> Enter time limit 3600 seconds for transaction	
1. The Charging Station sends a TransactionEventRequest	2. The Test System responds with a TransactionEventResponse
3. The Charging Station sends a TransactionEventRequest	4. The Test System responds with a TransactionEventResponse
<u>Manual Action:</u> Change time limit to 20 seconds	
<u>Note:</u> Steps 3 and 4 may repeat while entering new limit	
5. The Charging Station sends a TransactionEventRequest	6. The Test System responds with a TransactionEventResponse
7. The Charging Station sends a TransactionEventRequest	8. The Test System responds with a TransactionEventResponse
<u>Note:</u> Steps 7 and 8 may repeat until time reached	
9. The Charging Station sends a TransactionEventRequest	10. The Test System responds with a TransactionEventResponse
<u>Note:</u> Step 9 should be triggered when 20 seconds have elapsed since start of transaction	

Tool validations
<p>* Step 1:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - eventType must be <i>Updated</i> - triggerReason must be <i>LimitSet</i> - transactionInfo.transactionLimit.maxEnergy must be <i><omitted></i> - transactionInfo.transactionLimit.maxTime must be <i>3600</i> - transactionInfo.transactionLimit.maxCost must be <i><omitted></i> <p>* Step 3:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - eventType must be <i>Updated</i> - triggerReason must be <i>MeterValuePeriodic</i> <p>* Step 5:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - eventType must be <i>Updated</i> - triggerReason must be <i>LimitSet</i> - transactionInfo.transactionLimit.maxEnergy must be <i><omitted></i> - transactionInfo.transactionLimit.maxTime must be <i>20</i> - transactionInfo.transactionLimit.maxCost must be <i><omitted></i> <p>* Step 7:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - eventType must be <i>Updated</i> - triggerReason must be <i>MeterValuePeriodic</i> <p>* Step 9:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - eventType must be <i>Updated</i> (or <i>Ended</i> if TxStopPoint contains "EnergyTransfer") - triggerReason must be <i>TimeLimitReached</i> - transactionInfo.transactionLimit.maxEnergy must be <i><omitted></i> - transactionInfo.transactionLimit.maxTime must be <i>20</i> - transactionInfo.transactionLimit.maxCost must be <i><omitted></i> - transactionInfo.chargingState must be <i>SuspendedEVSE</i> (or <i>EVConnected</i> if TxStopPoint contains "EnergyTransfer")
<p>Post scenario validations:</p> <p>N/a</p>

TC_E_108_CS: Transactions with fixed cost, energy or time - CS calculates costs and specifies limit

Test case name	Transactions with fixed cost, energy or time - CS calculates costs and specifies limit
Test case Id	TC_E_108_CS
Use case Id(s)	E16
Requirement(s)	E16.FR.03, E16.FR.05, E16.FR.06
System under test	Charging Station
Description	CS will set a limit the transaction for the specified cost. CS will use local cost calculation.
Purpose	To verify whether the Charging Station correctly uses local cost calculation.
Prerequisite(s)	<ul style="list-style-type: none"> - Charging Station supports local cost calculation. - Charging Station supports manual entering of maximum cost. - TxCtrlr.SupportedLimits contains <i>maxCost</i>

Before (Preparations)

Configuration State:

TariffCostCtrlr.Enabled[Tariff] is *true*

TariffCostCtrlr.Enabled[RunningCost] is *true*

TariffCostCtrlr.Interval[Cost] is *60*

SampledDataCtrlr.TxUpdatedInterval is *0*

Memory State:

N/a

Reusable State(s):

State is *EVConnectedPreSession*

Main (Test scenario)

Charging Station	CSMS
2. The Charging Station responds with a SetDefaultTariffResponse	1. The Test System sends a SetDefaultTariffRequest with evseld <i>0</i> tariff.tariffId <i>Test System1</i> tariff.currency <i>EUR</i> tariff.chargingTime.prices[0].priceMinute <i>1.00</i> tariff.chargingTime.taxRates[0].type <i>"No Tax"</i> tariff.chargingTime.taxRates[0].tax <i>0.00</i> tariff.idleTime.prices[0].priceMinute <i>1.00</i> tariff.idleTime.taxRates[0].type <i>"No Tax"</i> tariff.idleTime.taxRates[0].tax <i>0.00</i>
3. Execute Reusable State <i>EnergyTransferStarted</i>	
4. The Charging Station sends a TransactionEventRequest	5. The Test System responds with a TransactionEventResponse
<u>Manual Action:</u> Enter cost limit 2.00 for transaction	
<u>Note:</u> Step 4 and 5 may be repeated while entering limit	
6. The Charging Station sends a TransactionEventRequest	7. The Test System responds with a TransactionEventResponse
8. The Charging Station sends a TransactionEventRequest	9. The Test System responds with a TransactionEventResponse
<u>Note:</u> Step 8 and 9 will be repeated until limit reached	
10. The Charging Station sends a TransactionEventRequest	11. The Test System responds with a TransactionEventResponse
<u>Note:</u> Step 10 should trigger automatically after 2 minutes have been elapsed since start of transaction	

Tool validations
<p>* Step 4:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - eventType must be <i>Updated</i> - triggerReason must be <i>RunningCost</i> - transactionInfo.transactionLimit must be <i><omitted></i> - costDetails.totalCost.currency must be <i>EUR</i> - costDetails.totalCost.typeOfCost must be <i>NormalCost</i> <p>* Step 6:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - eventType must be <i>Updated</i> - triggerReason must be <i>LimitSet</i> - transactionInfo.transactionLimit.maxEnergy must be <i><omitted></i> - transactionInfo.transactionLimit.maxTime must be <i><omitted></i> - transactionInfo.transactionLimit.maxCost must be <i>2.00</i> <p>* Step 8:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - eventType must be <i>Updated</i> - triggerReason must be <i>RunningCost</i> - costDetails.totalCost.currency must be <i>EUR</i> - costDetails.totalCost.typeOfCost must be <i>NormalCost</i> - costDetails.totalCost.total.exclTax must be <i>< 2.00</i> - costDetails.totalCost.total.inclTax must be <i>< 2.00</i>
<p>* Step 10:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - eventType must be <i>Updated</i> (or <i>Ended</i> if TxStopPoint contains "EnergyTransfer") - triggerReason must be <i>CostLimitReached</i> - transactionInfo.transactionLimit.maxEnergy must be <i><omitted></i> - transactionInfo.transactionLimit.maxTime must be <i><omitted></i> - transactionInfo.transactionLimit.maxCost must be <i>2.00</i> - costDetails.totalCost.currency must be <i>EUR</i> - costDetails.totalCost.typeOfCost must be <i>NormalCost</i> - costDetails.totalCost.total.exclTax must be <i>2.00</i> - costDetails.totalCost.total.inclTax must be <i>2.00</i> - transactionInfo.chargingState must be <i>SuspendedEVSE</i> (or <i>EVConnected</i> if TxStopPoint contains "EnergyTransfer")
<p>Post scenario validations:</p> <p>N/a</p>

TC_E_112_CS: Resuming transaction after interruption - TxResumptionTimeout not expired - TxAllowEnergyTransferResumption is false

Test case name	Resuming transaction after interruption - TxResumptionTimeout not expired - TxAllowEnergyTransferResumption is false
Test case Id	TC_E_112_CS
Use case Id(s)	E17
Requirement(s)	E17.FR.01, E17.FR.10, E17.FR.12, E17.FR.14
System under test	Charging Station
Description	Charging station is able to restore running transactions after a reboot.
Purpose	To verify whether the Charging Station was able to resume transactions after a reboot.
Prerequisite(s)	The Charging Station supports transaction resumption after a reboot.

Before (Preparations)
Configuration State: TxCtrlr.ResumptionTimeout is 999 TxCtrlr.AllowEnergyTransferResumption is false
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action:</u> Cut the power to the Charging Station	
<u>Manual Action:</u> Wait <Configured Transaction duration>	
<u>Manual Action:</u> Restore the power to the Charging Station	
1. The Charging Station sends a BootNotificationRequest	2. The Test System responds with a BootNotificationResponse with status Accepted
3. The Charging Station notifies the CSMS about the current state of all connectors. <i>It is allowed for the Charging Station to report status Unavailable as an intermediate status right before or after rebooting.</i> <i>It is allowed for the Charging Station to report status Available as an intermediate status for the <Configured EVSE.Connector></i>	4. The Test System responds accordingly.
5 The Charging Station sends a SecurityEventNotificationRequest Step 3 through 13 may be transmitted in any order.	6 The Test System responds with a SecurityEventNotificationResponse
7 The Charging Station sends a NotifyEventRequest Example of content error message: eventData.trigger = Delta eventData.transactionId = <transactionId from this transaction> eventData.eventNotificationType = HardWiredNotification eventData.severity = 3 eventData.actualValue = true eventData.component.name = ElectricalFeed eventData.variable.name = Problem <i>This step is optional. The Charging Station may report the power failure to the CSMS.</i>	8 The Test System responds with a NotifyEventResponse

Main (Test scenario)	
9 The Charging Station sends a NotifyEventRequest Same content as step 7, but with actualValue = false: eventData.actualValue = false <i>This step is optional: If having reported the problem at step 7, the Charging Station reports the problem has been resolved.</i>	10 The Test System responds with a NotifyEventResponse
11. The Charging Station sends a TransactionEventRequest <i>This step is optional: In some cases, a Charging Station might re-detect the cable being plugged in after a reboot.</i>	12. The Test System responds with a TransactionEventResponse
13. The Charging Station sends a TransactionEventRequest	14. The Test System responds with a TransactionEventResponse

Tool validations
<p>* Step 3:</p> <p>Message: StatusNotificationRequest</p> <ul style="list-style-type: none"> - evseld not 0 - connectorId not 0 <p>Configured EVSE.Connector:</p> <ul style="list-style-type: none"> - connectorStatus <i>Occupied</i> <p>Other connectors:</p> <ul style="list-style-type: none"> - connectorStatus <i>Available</i> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none"> - eventData[0].trigger <i>Delta</i> - eventData[0].component.name <i>"Connector"</i> - eventData[0].variable.name <i>"AvailabilityState"</i> <p>Configured EVSE.Connector:</p> <ul style="list-style-type: none"> - eventData[0].actualValue <i>"Occupied"</i> <p>Other connectors:</p> <ul style="list-style-type: none"> - eventData[0].actualValue <i>"Available"</i> <p>* Step 5:</p> <p>Message: SecurityEventNotificationRequest</p> <ul style="list-style-type: none"> - type must be <i>StartupOfTheDevice</i> OR <i>ResetOrReboot</i> <p>* Step 11:</p> <p>Message TransactionEventRequest</p> <ul style="list-style-type: none"> - eventType = <i>Updated</i> - triggerReason = <i>CablePluggedIn</i> - transactionInfo.transactionId <i><transactionId remained the same></i> <p>* Step 13:</p> <p>Message TransactionEventRequest</p> <ul style="list-style-type: none"> - eventType = <i>Updated</i> - triggerReason = <i>TxResumed</i> - transactionInfo.chargingState <i>SuspendedEVSE</i> - transactionInfo.transactionId <i><transactionId remained the same></i> <p>Post scenario validations: N/a</p>

TC_E_113_CS: Resuming transaction after interruption - TxResumptionTimeout not expired - TxAllowEnergyTransferResumption is true

Test case name	Resuming transaction after interruption - TxResumptionTimeout not expired - TxAllowEnergyTransferResumption is true
Test case Id	TC_E_113_CS
Use case Id(s)	E17
Requirement(s)	E17.FR.01, E17.FR.10, E17.FR.11, E17.FR.14
System under test	Charging Station
Description	Charging station is able to restore running transactions after a reboot.
Purpose	To verify whether the Charging Station was able to resume transactions after a reboot.
Prerequisite(s)	The Charging Station supports transaction and energy transfer resumption after a reboot.

Before (Preparations)
Configuration State: TxCtrlr.ResumptionTimeout is 999 TxCtrlr.AllowEnergyTransferResumption is true
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action:</u> Cut the power to the Charging Station	
<u>Manual Action:</u> Wait <Configured Transaction duration>	
<u>Manual Action:</u> Restore the power to the Charging Station	
1. The Charging Station sends a BootNotificationRequest	2. The Test System responds with a BootNotificationResponse with status Accepted
3. The Charging Station notifies the CSMS about the current state of all connectors. <i>It is allowed for the Charging Station to report status Unavailable as an intermediate status right before or after rebooting.</i> <i>It is allowed for the Charging Station to report status Available as an intermediate status for the <Configured EVSE.Connector></i>	4. The Test System responds accordingly.
5 The Charging Station sends a SecurityEventNotificationRequest Step 3 through 13 may be transmitted in any order.	6 The Test System responds with a SecurityEventNotificationResponse
7 The Charging Station sends a NotifyEventRequest Example of content error message: eventData.trigger = Delta eventData.transactionId = <transactionId from this transaction> eventData.eventNotificationType = HardWiredNotification eventData.severity = 3 eventData.actualValue = true eventData.component.name = ElectricalFeed eventData.variable.name = Problem <i>This step is optional. The Charging Station may report the power failure to the CSMS.</i>	8 The Test System responds with a NotifyEventResponse

Main (Test scenario)	
9 The Charging Station sends a NotifyEventRequest Same content as step 7, but with actualValue = false: eventData.actualValue = false <i>This step is optional: If having reported the problem at step 7, the Charging Station reports the problem has been resolved.</i>	10 The Test System responds with a NotifyEventResponse
11. The Charging Station sends a TransactionEventRequest <i>This step is optional: In some cases, a Charging Station might re-detect the cable being plugged in after a reboot.</i>	12. The Test System responds with a TransactionEventResponse
13. The Charging Station sends a TransactionEventRequest	14. The Test System responds with a TransactionEventResponse

Tool validations
<p>* Step 3:</p> <p>Message: StatusNotificationRequest</p> <ul style="list-style-type: none"> - evseld not 0 - connectorId not 0 <p>Configured EVSE.Connector:</p> <ul style="list-style-type: none"> - connectorStatus <i>Occupied</i> <p>Other connectors:</p> <ul style="list-style-type: none"> - connectorStatus <i>Available</i> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none"> - eventData[0].trigger <i>Delta</i> - eventData[0].component.name <i>"Connector"</i> - eventData[0].variable.name <i>"AvailabilityState"</i> <p>Configured EVSE.Connector:</p> <ul style="list-style-type: none"> - eventData[0].actualValue <i>"Occupied"</i> <p>Other connectors:</p> <ul style="list-style-type: none"> - eventData[0].actualValue <i>"Available"</i> <p>* Step 5:</p> <p>Message: SecurityEventNotificationRequest</p> <ul style="list-style-type: none"> - type must be <i>StartupOfTheDevice</i> OR <i>ResetOrReboot</i> <p>* Step 11:</p> <p>Message TransactionEventRequest</p> <ul style="list-style-type: none"> - eventType = <i>Updated</i> - triggerReason = <i>CablePluggedIn</i> - transactionInfo.transactionId <i><transactionId remained the same></i> <p>* Step 13:</p> <p>Message TransactionEventRequest</p> <ul style="list-style-type: none"> - eventType = <i>Updated</i> - triggerReason = <i>TxResumed</i> - transactionInfo.chargingState <i>Charging</i> - transactionInfo.transactionId <i><transactionId remained the same></i> <p>Post scenario validations: N/a</p>

TC_E_114_CS: Resuming transaction after interruption - Powerloss - TxResumptionTimeout absent

Test case name	Resuming transaction after interruption - Powerloss - TxResumptionTimeout absent
Test case Id	TC_E_114_CS
Use case Id(s)	E17
Requirement(s)	E17.FR.01
System under test	Charging Station
Description	Charging station does not support resuming the transactions that were running before powerloss.
Purpose	To verify whether the Charging Station does not resume the transactions.
Prerequisite(s)	TxCtrlr.ResumptionTimeout is <i><absent></i>

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action:</u> Cut the power to the Charging Station, wait 30 seconds and restore power to the Charging Station	

Tool validations
N/a
Post scenario validations: Charging Station boots normally, but no Transaction related messages are sent to the CSMS with <i>triggerReason</i> = TxResumed, since resuming is not supported. <u>Note</u> : It is allowed that some TransactionEventRequests are sent to report that the running transactions have been ended, e.g. via <i>triggerReason</i> = AbnormalCondition.

TC_E_115_CS: Resuming transaction after interruption - Powerloss - TxResumptionTimeout = 0

Test case name	Resuming transaction after interruption - Powerloss - TxResumptionTimeout = 0
Test case Id	TC_E_115_CS
Use case Id(s)	E17
Requirement(s)	E17.FR.01, E17.FR.20, E17.FR.21, E18.FR.22
System under test	Charging Station
Description	Charging station is configured to not resume the transactions that were running before powerloss.
Purpose	To verify whether the Charging Station does not resume the transactions.
Prerequisite(s)	

Before (Preparations)
Configuration State: TxCtrlr.ResumptionTimeout is 0
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferStarted</i>

Main (Test scenario)
Charging Station CSMS
Manual Action: Cut the power to the Charging Station, wait 30 seconds and restore power to the Charging Station

Tool validations
N/a
Post scenario validations: Charging Station boots normally, and TransactionEventRequests with <i>eventType</i> = Ended and <i>triggerReason</i> = AbnormalCondition are sent to report that the running transactions have been ended. <i>stoppedReason</i> must be either <i>PowerLoss</i> or <i>Reboot</i> .

TC_E_116_CS: Resuming transaction after interruption - Powerloss - TxResumptionTimeout expired

Test case name	Resuming transaction after interruption - Powerloss - TxResumptionTimeout expired
Test case Id	TC_E_116_CS
Use case Id(s)	E17
Requirement(s)	E17.FR.01, E17.FR.20, E17.FR.21
System under test	Charging Station
Description	Charging station reports running transactions as ended after a reboot and TxResumptionTimeout occurred.
Purpose	To verify whether the Charging Station was able to end the transactions after a reboot and an expired TxResumptionTimeout.
Prerequisite(s)	

Before (Preparations)
Configuration State: TxCtrlr.ResumptionTimeout is <Configured Transaction duration>
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action:</u> Cut the power to the Charging Station	
<u>Manual Action:</u> Wait <Configured Transaction duration + 1> seconds	
<u>Manual Action:</u> Restore the power to the Charging Station	
1. The Charging Station sends a BootNotificationRequest	2. The Test System responds with a BootNotificationResponse with status <i>Accepted</i>
3. The Charging Station notifies the CSMS about the current state of all connectors. <i>It is allowed for the Charging Station to report status Unavailable as an intermediate status right before or after rebooting.</i> <i>It is allowed for the Charging Station to report status Available as an intermediate status for the <Configured EVSE.Connector></i>	4. The Test System responds accordingly.
5 The Charging Station sends a SecurityEventNotificationRequest Step 3 through 13 may be transmitted in any order.	6 The Test System responds with a SecurityEventNotificationResponse
7 The Charging Station sends a NotifyEventRequest Example of content error message: eventData.trigger = <i>Delta</i> eventData.transactionId = <transactionId from this transaction> eventData.eventNotificationType = <i>HardWiredNotification</i> eventData.severity = 3 eventData.actualValue = <i>true</i> eventData.component.name = <i>ElectricalFeed</i> eventData.variable.name = <i>Problem</i> <i>This step is optional. The Charging Station may report the power failure to the CSMS.</i>	8 The Test System responds with a NotifyEventResponse

Main (Test scenario)	
9 The Charging Station sends a NotifyEventRequest Same content as step 7, but with actualValue = false: eventData.actualValue = false <i>This step is optional: If having reported the problem at step 7, the Charging Station reports the problem has been resolved.</i>	10 The Test System responds with a NotifyEventResponse
11. The Charging Station sends a TransactionEventRequest	12. The Test System responds with a TransactionEventResponse
13. The Charging Station sends a TransactionEventRequest <i>This step is optional: In some cases, a Charging Station might re-detect the cable being plugged in after a reboot. In this case that is only possible when TxStartPoint contains EVConnected.</i>	14. The Test System responds with a TransactionEventResponse

Tool validations
<p>* Step 3:</p> <p>Message: StatusNotificationRequest</p> <ul style="list-style-type: none"> - evseld not 0 - connectorId not 0 <p>Configured EVSE.Connector:</p> <ul style="list-style-type: none"> - connectorStatus <i>Occupied</i> <p>Other connectors:</p> <ul style="list-style-type: none"> - connectorStatus <i>Available</i> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none"> - eventData[0].trigger <i>Delta</i> - eventData[0].component.name <i>"Connector"</i> - eventData[0].variable.name <i>"AvailabilityState"</i> <p>Configured EVSE.Connector:</p> <ul style="list-style-type: none"> - eventData[0].actualValue <i>"Occupied"</i> <p>Other connectors:</p> <ul style="list-style-type: none"> - eventData[0].actualValue <i>"Available"</i> <p>* Step 5:</p> <p>Message: SecurityEventNotificationRequest</p> <ul style="list-style-type: none"> - type must be <i>StartupOfTheDevice</i> OR <i>ResetOrReboot</i> <p>* Step 11:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - eventType must be <i>Ended</i> - triggerReason must be <i>AbnormalCondition</i> - stoppedReason must be <i>PowerLoss</i> <p>* Step 13:</p> <p>Message TransactionEventRequest</p> <ul style="list-style-type: none"> - eventType = <i>Started</i> - triggerReason = <i>CablePluggedIn</i> or <i>ChargingStateChanged</i> - transactionInfo.chargingState <i>EVConnected</i> - transactionInfo.transactionId <i><Different transactionId than previous transaction></i> <p>Post scenario validations: N/a</p>

F Remote Control

TC_F_01_CS: Remote start transaction - Cable plugin first

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 <i>true</i> (If implemented AND ReadWrite) AuthCtrlr.DisableRemoteAuthorization is <i>false</i> (If implemented)
Memory State: N/a
Reusable State(s): State is <i>EVConnectedPreSession</i>

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>Authorized</i> (remote)	
2. Execute Reusable State <i>EnergyTransferStarted</i>	

Tool validations
N/a
Post scenario validations: N/a

TC_F_02_CS: Remote start transaction - Remote start first - AuthorizeRemoteStart is true

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 <i>true</i> (If implemented AND ReadWrite) AuthCtrlr.DisableRemoteAuthorization is <i>false</i> (If implemented) AuthorizeRemoteStart is <i>true</i> (If ReadWrite)
Memory State: N/a
Reusable State(s): State is <i>ParkingBayOccupied</i> (Optional state)

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>Authorized</i> (remote)	
2. Execute Reusable State <i>EnergyTransferStarted</i>	

Tool validations
N/a
Post scenario validations: N/a

TC_F_03_CS: Remote start transaction - Remote start first - AuthorizeRemoteStart is false

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 <i>true</i> (If implemented AND ReadWrite) AuthCtrlr.DisableRemoteAuthorization is <i>false</i> (If implemented) AuthorizeRemoteStart is <i>false</i> (If ReadWrite)
Memory State: N/a
Reusable State(s): State is <i>ParkingBayOccupied</i> (Optional state)

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>Authorized</i> (remote)	
2. Execute Reusable State <i>EnergyTransferStarted</i>	

Tool validations
N/a
Post scenario validations: N/a

TC_F_04_CS: Remote start transaction - Remote start first - Cable plugin timeout

Test case name	Remote start transaction - Remote start first - Cable plugin timeout
Test case Id	TC_F_04_CS
Use case Id(s)	F02, E03
Requirement(s)	F02.FR.01, E03.FR.01, E03.FR.05
System under test	Charging Station
Description	OCPP 2.x.x allows an EV driver to either first wait for/trigger a RequestStartTransactionRequest OR connect the EV and EVSE. Both sequences will result in being able to charge.
Purpose	To verify if the Charging Station is able to deauthorize the transaction after the EVConnectionTimeout has been reached.
Prerequisite(s)	N/a

Before (Preparations)

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* (If supported)

Memory State:

N/a

Reusable State(s):

State is *Authorized* (remote)

Main (Test scenario)

Charging Station	CSMS
1. The Charging Station sends a TransactionEventRequest <u>Note(s):</u> - 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 <i>ParkingBayOccupancy</i> OR <i>Authorized</i>	2. The Test System responds with a TransactionEventResponse
<u>Note(s):</u> Optionally the Charging Station can send a StatusNotificationRequest or NotifyEventRequest with status Available	
3. Execute Reusable State <i>Authorized</i> (remote) <u>Note(s):</u> - 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

TC_F_05_CS: Remote unlock Connector - With ongoing transaction

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
Reusable 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 Test System sends a UnlockConnectorRequest with evseld <Configured evseld> connectorId <Configured connectorId>

Tool validations
* Step 2: Message UnlockConnectorResponse - status <i>OngoingAuthorizedTransaction</i>
Post scenario validations: - N/a

TC_F_06_CS: Remote unlock Connector - Without ongoing transaction - Accepted

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 send 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

Reusable State:

N/a

Main (Test scenario)

Charging Station	CSMS
2. The Charging Station responds with a UnlockConnectorResponse	1. The Test System sends a UnlockConnectorRequest with evseld <Configured evseld> connectorId <Configured connectorId>

Tool validations

* Step 2:

Message **UnlockConnectorResponse**- **status** *Unlocked*

Post scenario validations:

- N/a

TC_F_07_CS: Remote unlock Connector - Without ongoing transaction - No cable connected

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 send 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 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
Reusable State: No cable connected at <Configured Connector>

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a UnlockConnectorResponse	1. The Test System sends a UnlockConnectorRequest with evseld <Configured evseld> connectorId <Configured connectorId>

Tool validations
* Step 2: Message UnlockConnectorResponse - status <i>Unlocked</i>
Post scenario validations: - N/a

TC_F_08_CS: Remote stop transaction - Success

Test case name	Remote stop transaction - Success
Test case Id	TC_F_08_CS
Use case Id(s)	F03
Requirement(s)	F03.FR.02, F03.FR.03, F03.FR.07, F03.FR.09
System under test	Charging Station
Description	The CSMS is able to stop a charging session remotely by sending a RequestStopTransactionRequest to the Charging Station.
Purpose	To verify if the Charging Station is able to stop a charging session when it receives a RequestStopTransactionRequest message.
Prerequisite(s)	N/a

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>StopAuthorized</i> (remote)	

Tool validations
N/a
Post scenario validations: N/a

TC_F_09_CS: Remote stop transaction - Rejected

Test case name	Remote stop transaction - Rejected
Test case Id	TC_F_09_CS
Use case Id(s)	F03
Requirement(s)	F03.FR.08
System under test	Charging Station
Description	The CSMS is able to stop a charging session remotely by sending a RequestStopTransactionRequest to the Charging Station.
Purpose	To verify if the Charging Station will reject a RequestStopTransactionRequest message, if it contains a transactionId that cannot be matched to an active transaction.
Prerequisite(s)	N/a

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a RequestStopTransactionResponse	1. The Test System 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

TC_F_10_CS: Remote unlock Connector - Without ongoing transaction - UnknownConnector

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 send an UnlockConnectorRequest to the charging station. It sometimes happens that a connector of a Charging Station socket does not unlock correctly. This happens most of the time when there is tension on the charging cable. This means the driver cannot unplug his charging cable from the Charging Station. To help a driver, the CSO can send a UnlockConnectorRequest to the Charging Station. The Charging Station will then try to unlock the connector again.
Purpose	To verify if the Charging Station is able to respond with a UnlockConnectorRequest with status <i>UnknownConnector</i> when the requested connector is unknown as described in the OCPP specification.
Prerequisite(s)	The Charging Station has a connector lock.

Before (Preparations)

Configuration State:

N/a

Memory State:

N/a

Reusable State:

N/a

Main (Test scenario)

Charging Station	CSMS
2. The Charging Station responds with a UnlockConnectorResponse	1. The Test System sends a UnlockConnectorRequest with evseld <Configured evseld> connectorId 999

Tool validations

* Step 2:

Message **UnlockConnectorResponse**- **status** *UnknownConnector*

Post scenario validations:

- N/a

TC_F_11_CS: Trigger message - MeterValues - Specific EVSE

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 Test System sends a TriggerMessageRequest With requestedMessage <i>MeterValues</i> evse.id <Configured evseld>
3. The Charging Station sends a MeterValuesRequest	4. The Test System responds with a MeterValuesResponse

Tool validations
<p>* Step 2: Message: TriggerMessageResponse - status must be <i>Accepted</i></p> <p>* Step 3: Message: MeterValuesRequest - evseld must be <Configured evseld> - meterValue[0].sampledValue[0].context must be <i>Trigger</i></p>
Post scenario validations: N/a

TC_F_12_CS: Trigger message - MeterValues - All EVSE

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 Test System sends a TriggerMessageRequest With requestedMessage <i>MeterValues</i> evse is omitted
3. The Charging Station sends a MeterValuesRequest	4. The Test System responds with a MeterValuesResponse
<u>Note(s):</u> - This step needs to be executed for every EVSE.	

Tool validations
* Step 2: Message: TriggerMessageResponse - status must be <i>Accepted</i> * Step 3: Message: MeterValuesRequest - meterValue[0].sampledValue[0].context must be <i>Trigger</i>
Post scenario validations: N/a

TC_F_13_CS: Trigger message - TransactionEvent - Specific EVSE

Test case name	Trigger message - TransactionEvent - Specific EVSE
Test case Id	TC_F_13_CS
Use case Id(s)	F06
Requirement(s)	F06.FR.02,F06.FR.04,F06.FR.05,F06.FR.07,F06.FR.10
System under test	Charging Station
Description	The CSMS can request a Charging Station to send Charging Station-initiated messages. In the request the CSMS indicates which message it wishes to receive.
Purpose	To verify if the Charging Station is able to send a TransactionEventRequest message for a specific EVSE, after receiving a TriggerMessageRequest message.
Prerequisite(s)	The Charging Station supports sending TransactionEvents triggered by a TriggerMessageRequest.

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a TriggerMessageResponse	1. The Test System sends a TriggerMessageRequest With requestedMessage <i>TransactionEvent</i> evse.id <Configured evseld>
3. The Charging Station sends a TransactionEventRequest	4. The Test System responds with a TransactionEventResponse

Tool validations
<p>* Step 2: Message: TriggerMessageResponse - status must be <i>Accepted</i></p> <p>* Step 3: Message: TransactionEventRequest - evse.id must be <i>omitted</i> or <Configured evseld> - triggerReason must be <i>Trigger</i> - transactionInfo.chargingState must be <i>Charging</i> - meterValue must be present - meterValue[0].sampledValue[0].context must be <i>Trigger</i></p> <p>Post scenario validations: N/a</p>

TC_F_14_CS: Trigger message - TransactionEvent - All EVSE

Test case name	Trigger message - TransactionEvent - All EVSE
Test case Id	TC_F_14_CS
Use case Id(s)	F06
Requirement(s)	F06.FR.04,F06.FR.05,F06.FR.07,F06.FR.10,F06.FR.11
System under test	Charging Station
Description	The CSMS can request a Charging Station to send Charging Station-initiated messages. In the request the CSMS indicates which message it wishes to receive.
Purpose	To verify if the Charging Station is able to send a TransactionEventRequest message for all EVSE, after receiving a TriggerMessageRequest message.
Prerequisite(s)	The Charging Station supports sending TransactionEvents triggered by a TriggerMessageRequest.

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferStarted</i> for all EVSE

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a TriggerMessageResponse	1. The Test System sends a TriggerMessageRequest With requestedMessage <i>TransactionEvent</i> evse is omitted
3. The Charging Station sends a TransactionEventRequest	4. The Test System responds with a TransactionEventResponse
<u>Note(s):</u> - This step needs to be executed for every EVSE.	

Tool validations
<p>* Step 2: Message: TriggerMessageResponse - status must be <i>Accepted</i></p> <p>* Step 3: Message: TransactionEventRequest - evse.id must be <i><Configured evseld></i> - triggerReason must be <i>Trigger</i> - transactionInfo.chargingState must be <i>Charging</i> - meterValue must be present - meterValue[0].sampledValue[0].context must be <i>Trigger</i></p>
Post scenario validations: N/a

TC_F_15_CS: Trigger message - LogStatusNotification - Idle

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 Test System sends a TriggerMessageRequest With requestedMessage <i>LogStatusNotification</i>
3. The Charging Station sends a LogStatusNotificationRequest	4. The Test System responds with a LogStatusNotificationResponse

Tool validations
<p>* Step 2: Message: TriggerMessageResponse - status must be <i>Accepted</i></p> <p>* Step 3: Message: LogStatusNotificationRequest - status must be <i>Idle</i></p>
Post scenario validations: N/a

TC_F_16_CS: Trigger message - LogStatusNotification - Uploading

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 Test System sends a GetLogRequest With logType <i>DiagnosticsLog</i> log.remoteLocation is <Configured log_location>
3. The Charging Station sends a LogStatusNotificationRequest	4. The Test System responds with a LogStatusNotificationResponse
6. The Charging Station responds with a TriggerMessageResponse	5. The Test System sends a TriggerMessageRequest With requestedMessage <i>LogStatusNotification</i>
7. The Charging Station sends a LogStatusNotificationRequest	8. The Test System 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

TC_F_17_CS: Trigger message - FirmwareStatusNotification - Specific EVSE not relevant

Test case name	Trigger message - FirmwareStatusNotification - Specific EVSE not relevant
Test case Id	TC_F_17_CS
Use case Id(s)	F06
Requirement(s)	F06.FR.03,F06.FR.04,F06.FR.05,F06.FR.10
System under test	Charging Station
Description	The CSMS can request a Charging Station to send Charging Station-initiated messages. In the request the CSMS indicates which message it wishes to receive.
Purpose	To verify if the Charging Station is able to send a FirmwareStatusNotificationRequest, after receiving a TriggerMessageRequest even when the CSMS an evseld which is not relevant for the requestedMessage FirmwareStatusNotification.
Prerequisite(s)	The Charging Station supports sending FirmwareStatusNotifications triggered by a TriggerMessageRequest.

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

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a TriggerMessageResponse	1. The Test System sends a TriggerMessageRequest With requestedMessage <i>FirmwareStatusNotification</i> evse.id is <Configured evseld>
3. The Charging Station sends a FirmwareStatusNotificationRequest	4. The Test System responds with a FirmwareStatusNotificationResponse

Tool validations
<p>* Step 2: Message: TriggerMessageResponse - status must be <i>Accepted</i></p> <p>* Step 3: Message: FirmwareStatusNotificationRequest - status must be <i>Idle</i></p>
Post scenario validations: N/a

TC_F_18_CS: Trigger message - FirmwareStatusNotification - Idle

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 Test System sends a TriggerMessageRequest With requestedMessage <i>FirmwareStatusNotification</i>
3. The Charging Station sends a FirmwareStatusNotificationRequest	4. The Test System responds with a FirmwareStatusNotificationResponse

Tool validations
<p>* Step 2: Message: TriggerMessageResponse - status must be <i>Accepted</i></p> <p>* Step 3: Message: FirmwareStatusNotificationRequest - status must be <i>Idle</i></p>
Post scenario validations: N/a

TC_F_19_CS: Trigger message - FirmwareStatusNotification - Downloading

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 Test System 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 invalid firmware signature>
3. The Charging Station sends a FirmwareStatusNotificationRequest	4. The Test System responds with a FirmwareStatusNotificationResponse
6. The Charging Station responds with a TriggerMessageResponse	5. The Test System sends a TriggerMessageRequest With requestedMessage <i>FirmwareStatusNotification</i>
7. The Charging Station sends a FirmwareStatusNotificationRequest	8. The Test System responds with a FirmwareStatusNotificationResponse
<i>Note: Step 9 through 14 are cleanup to prevent an ongoing firmware update after the testcase is already ended. The behavior part of these steps is part of TC_L_06_CS and therefore not part of the scope for this testcase.</i>	
9. The Charging Station sends a FirmwareStatusNotificationRequest . With status Downloaded	10. The Test System responds with a FirmwareStatusNotificationResponse .
11. The Charging Station sends a FirmwareStatusNotificationRequest . With status InvalidSignature	12. The Test System responds with a FirmwareStatusNotificationResponse .
13. The Charging Station sends a SecurityEventNotificationRequest . With type InvalidFirmwareSignature	14. The Test System responds with a SecurityEventNotificationResponse .

Tool validations
<p>* Step 2: Message: UpdateFirmwareResponse - status must be <i>Accepted</i></p> <p>* Step 3: Message: FirmwareStatusNotificationRequest - status must be <i>Downloading</i></p> <p>* Step 6: Message: TriggerMessageResponse - status must be <i>Accepted</i></p> <p>* Step 7: Message: FirmwareStatusNotificationRequest - status must be <i>Downloading</i></p>
<p>Post scenario validations: N/a</p>

TC_F_20_CS: Trigger message - Heartbeat

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 Test System sends a TriggerMessageRequest With requestedMessage Heartbeat
3. The Charging Station sends a HeartbeatRequest	4. The Test System responds with a HeartbeatResponse

Tool validations
* Step 2: Message: TriggerMessageResponse - status must be <i>Accepted</i>
Post scenario validations: N/a

TC_F_23_CS: Trigger message - StatusNotification - Specific EVSE - Available

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 Test System sends a TriggerMessageRequest With requestedMessage <i>StatusNotification</i> evse.id <Configured evseId> evse.connectorId <Configured connectorId>
3. The Charging Station notifies the CSMS about the current state of the connector.	4. The Test System 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

TC_F_24_CS: Trigger message - StatusNotification - Specific EVSE - Occupied

Test case name	Trigger message - StatusNotification - Specific EVSE - Occupied
Test case Id	TC_F_24_CS
Use case Id(s)	F06
Requirement(s)	F06.FR.02,F06.FR.04,F06.FR.05,F06.FR.10
System under test	Charging Station
Description	The CSMS can request a Charging Station to send Charging Station-initiated messages. In the request the CSMS indicates which message it wishes to receive.
Purpose	To verify if the Charging Station is able to send a StatusNotificationRequest message for a specific occupied EVSE/Connector, after receiving a TriggerMessageRequest message.
Prerequisite(s)	The Charging Station supports sending StatusNotifications triggered by a TriggerMessageRequest.

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>EVConnectedPreSession</i>

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a TriggerMessageResponse	1. The Test System sends a TriggerMessageRequest With requestedMessage <i>StatusNotification</i> evse.id <Configured evseld> evse.connectorId <Configured connectorId>
3. The Charging Station notifies the CSMS about the current state of the connector.	4. The Test System responds accordingly.

Tool validations
<p>* Step 2: Message: TriggerMessageResponse - status must be <i>Accepted</i></p> <p>* Step 3: Message: StatusNotificationRequest - connectorStatus <i>Occupied</i></p> <p>Message: NotifyEventRequest - eventData[0].trigger <i>Delta</i> - eventData[0].actualValue <i>"Occupied"</i> - eventData[0].component.name <i>"Connector"</i> - eventData[0].variable.name <i>"AvailabilityState"</i></p>
Post scenario validations: N/a

TC_F_26_CS: Trigger message - BootNotification - Rejected

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 Test System sends a TriggerMessageRequest With requestedMessage <i>BootNotification</i>

Tool validations
* Step 2: Message: TriggerMessageResponse - status must be <i>Rejected</i>
Post scenario validations: N/a

TC_F_27_CS: Trigger message - NotImplemented

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 Test System sends a TriggerMessageRequest With requestedMessage <i>SignCombinedCertificate</i>

Tool validations
* Step 2: Message: TriggerMessageResponse - status must be <i>NotImplemented</i>
Post scenario validations: N/a

TC_F_100_CS: Trigger message - CustomTrigger

Test case name	Trigger message - CustomTrigger
Test case Id	TC_F_100_CS
Use case Id(s)	F06
Requirement(s)	F06.FR.04, F06.FR.10, F06.FR.18, F06.FR.19
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 message corresponding to a custom trigger it reports to support.
Prerequisite(s)	The Charging Station supports custom triggers (CustomizationCtrlr.CustomTriggers exists and contains at least one value).

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 GetVariablesResponse	1. Test System sends GetVariablesRequest with - getVariableData[0].component.name = "CustomizationCtrlr" - getVariableData[0].variable.name = "CustomTriggers"
4. The Charging Station responds with a TriggerMessageResponse	3. The Test System sends a TriggerMessageRequest with requestedMessage CustomTrigger customTrigger <first mentioned in GetVariablesResponse.getVariableResult[0].attributeValue of step 2>
<u>Note:</u> The Charging Station send something based on the TriggerMessageRequest.	
6. The Charging Station responds with a TriggerMessageResponse	5. The Test System sends a TriggerMessageRequest with requestedMessage Heartbeat customTrigger <first mentioned in GetVariablesResponse.getVariableResult[0].attributeValue of step 2>
7. The Charging Station sends a HeartbeatRequest	8. The Test System responds with a HeartbeatResponse
10. The Charging Station responds with a TriggerMessageResponse	9. The Test System sends a TriggerMessageRequest with requestedMessage CustomTrigger customTrigger <generated random string value>

Tool validations
<p>* Step 2:</p> <p>Message: GetVariablesResponse</p> <ul style="list-style-type: none">- getVariableResult[0].attributeValue must be <i><not omitted></i>- getVariableResult[0].component.name must be <i>"CustomizationCtrlr"</i>- getVariableResult[0].variable.name must be <i>"CustomTriggers"</i> <p>* Step 4:</p> <p>Message: TriggerMessageResponse</p> <ul style="list-style-type: none">- status must be <i>Accepted</i> <p>* Step 6:</p> <p>Message: TriggerMessageResponse</p> <ul style="list-style-type: none">- status must be <i>Accepted</i> <p>* Step 10:</p> <p>Message: TriggerMessageResponse</p> <ul style="list-style-type: none">- status must be <i>NotImplemented</i>
<p>Post scenario validations:</p> <p>N/a</p>

G Availability**TC_G_01_CS: Connector status Notification - Available to Occupied**

Test case name	Connector status Notification - Available to Occupied
Test case Id	TC_G_01_CS
Use case Id(s)	G01, N07
Requirement(s)	G01.FR.01, N07.FR.19
System under test	Charging Station
Description	A Charging Station sends a notification to the CSMS to inform the CSMS about a Connector status change. This can be done in two ways. Via a StatusNotificationRequest or a NotifyEventRequest from the device model.
Purpose	To verify whether the Charging Station is able to report that its connector is <i>Occupied</i> .
Prerequisite(s)	N/a

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

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>EVConnectedPreSession</i>	

Tool validations
N/a
Post scenario validations: N/a

TC_G_02_CS: Connector status Notification - Occupied to Available

Test case name	Connector status Notification - Occupied to Available
Test case Id	TC_G_02_CS
Use case Id(s)	G01, N07
Requirement(s)	G01.FR.01, N07.FR.19
System under test	Charging Station
Description	A Charging Station sends a notification to the CSMS to inform the CSMS about a Connector status change. This can be done in two ways. Via a StatusNotificationRequest or a NotifyEventRequest from the device model.
Purpose	To verify whether the Charging Station is able to report that its connector is Available_.
Prerequisite(s)	N/a

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>EVConnectedPreSession</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action</u> : <i>Disconnect the EV and EVSE.</i>	
3. The Charging Station notifies the CSMS about the current state of the connector.	4. The Test System responds accordingly.

Tool validations
<p>* Step 3:</p> <p>Message: StatusNotificationRequest</p> <ul style="list-style-type: none"> - connectorStatus <i>Available</i> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none"> - eventData[0].trigger <i>Delta</i> - eventData[0].actualValue <i>"Available"</i> - eventData[0].component.name <i>"Connector"</i> - eventData[0].variable.name <i>"AvailabilityState"</i>
Post scenario validations: N/a

TC_G_03_CS: Change Availability EVSE - Operative to inoperative

Test case name	Change Availability EVSE - Operative to inoperative
Test case Id	TC_G_03_CS
Use case Id(s)	G03
Requirement(s)	G03.FR.01, G03.FR.02, G03.FR.04, G03.FR.06
System under test	Charging Station
Description	This test case covers how the CSMS requests the Charging Station to change the availability of one of the EVSEs from Operative to Inoperative. An EVSE is considered Operative in any status other than Faulted and Unavailable.
Purpose	To verify if the Charging Station is able to perform the change availability mechanism as described at the OCPP specification.
Prerequisite(s)	n/a

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

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>Unavailable</i> for <Configured evseld>	

Tool validations
N/a
Post scenario validations: - A message to report the state of a connector has been received for all connectors belonging to the specified EVSE.

TC_G_04_CS: Change Availability EVSE - Inoperative to operative

Test case name	Change Availability EVSE - Inoperative to operative
Test case Id	TC_G_04_CS
Use case Id(s)	G03
Requirement(s)	G03.FR.01, G03.FR.02, G03.FR.04, G03.FR.07
System under test	Charging Station
Description	This test case covers how the CSMS requests the Charging Station to change the availability of one of the EVSEs from Inoperative to Operative. An EVSE is considered Operative in any status other than Faulted and Unavailable.
Purpose	To verify if the Charging Station is able to perform the change availability mechanism as described at the OCPP specification.
Prerequisite(s)	n/a

Before (Preparations)
Configuration State: N/a
Memory State: <i>Unavailable</i> for <Configured evseId>
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a ChangeAvailabilityResponse	1. The Test System sends a ChangeAvailabilityRequest with operationalStatus <i>Operative</i> and evse.id <i><Configured evseId></i>
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 Test System responds accordingly.

Tool validations
<p>* Step 2: Message ChangeAvailabilityResponse - status <i>Accepted</i></p> <p>* Step 3: Message: StatusNotificationRequest - connectorStatus <i>Available</i> - evseId <Configured evseId></p> <p>Message: NotifyEventRequest - eventData[0].trigger <i>Delta</i> - eventData[0].actualValue <i>"Available"</i> - eventData[0].component.name <i>"EVSE" / Connector</i> - eventData[0].component.evse.id <Configured evseId> - eventData[0].variable.name <i>"AvailabilityState"</i></p>
<p>Post scenario validations: - A message to report the state of a connector has been received for all connectors belonging to the specified EVSE.</p>

TC_G_05_CS: Change Availability Charging Station - Operative to inoperative

Test case name	Change Availability Charging Station - Operative to inoperative
Test case Id	TC_G_05_CS
Use case Id(s)	G04
Requirement(s)	G04.FR.01, G04.FR.02, G04.FR.03, G04.FR.05, G04.FR.07
System under test	Charging Station
Description	<p>This test case describes how the CSMS requests the Charging Station to change the availability from operative to inoperative.</p> <p>A Charging Station is considered Operative when it is charging or ready for charging.</p> <p>A Charging Station is considered Inoperative when it does not allow any charging.</p>
Purpose	To verify if the Charging Station is able to perform the change availability mechanism as described at the OCPP specification.
Prerequisite(s)	n/a

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

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>Unavailable</i>	

Tool validations
N/a
Post scenario validations: - A message to report the state of a connector has been received for all connectors.

TC_G_06_CS: Change Availability Charging Station - Inoperative to operative

Test case name	Change Availability Charging Station - Inoperative to operative
Test case Id	TC_G_06_CS
Use case Id(s)	G04
Requirement(s)	G04.FR.01, G04.FR.02, G04.FR.03, G04.FR.05, G04.FR.08
System under test	Charging Station
Description	<p>This test case describes how the CSMS requests the Charging Station to change the availability from inoperative to operative.</p> <p>A Charging Station is considered Operative when it is charging or ready for charging.</p> <p>A Charging Station is considered Inoperative when it does not allow any charging.</p>
Purpose	To verify if the Charging Station is able to perform the change availability mechanism as described at the OCPP specification.
Prerequisite(s)	n/a

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>Unavailable</i>

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a ChangeAvailabilityResponse	1. The Test System sends a ChangeAvailabilityRequest with operationalStatus <i>Operative</i>
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 Test System responds accordingly.

Tool validations
<p>* Step 2: Message ChangeAvailabilityResponse - status <i>Accepted</i></p> <p>* Step 3: Message: StatusNotificationRequest - connectorStatus <i>Available</i> Message: NotifyEventRequest - eventData[0].trigger <i>Delta</i> - eventData[0].actualValue <i>"Available"</i> - eventData[0].component.name <i>"ChargingStation" / EVSE / Connector</i> - eventData[0].variable.name <i>"AvailabilityState"</i></p>
Post scenario validations: - A message to report the state of a connector has been received for all connectors.

TC_G_07_CS: Change Availability Connector - Operative to inoperative

Test case name	Change Availability Connector - Operative to inoperative
Test case Id	TC_G_07_CS
Use case Id(s)	G03
Requirement(s)	G03.FR.01, G03.FR.02, G03.FR.04, G03.FR.06
System under test	Charging Station
Description	This test case covers how the CSMS requests the Charging Station to change the availability of one of the Connectors of one EVSE from Operative to Inoperative. A Connector is considered Operative in any status other than Faulted and Unavailable.
Purpose	To verify if the Charging Station is able to perform the change availability mechanism as described at the OCPP specification.
Prerequisite(s)	n/a

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

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>Unavailable</i> for <Configured connectorId>	

Tool validations
N/a
Post scenario validations: - A message to report the state of the connector has been received.

TC_G_08_CS: Change Availability Connector - Inoperative to operative

Test case name	Change Availability Connector - Inoperative to operative
Test case Id	TC_G_08_CS
Use case Id(s)	G03
Requirement(s)	G03.FR.01, G03.FR.02, G03.FR.04, G03.FR.07
System under test	Charging Station
Description	This test case covers how the CSMS requests the Charging Station to change the availability of one of the Connectors from one EVSE from Inoperative to Operative. A Connector is considered Operative in any status other than Faulted and Unavailable.
Purpose	To verify if the Charging Station is able to perform the change availability mechanism as described at the OCPP specification.
Prerequisite(s)	n/a

Before (Preparations)
Configuration State: N/a
Memory State: <i>Unavailable</i> for <Configured connectorId>
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a ChangeAvailabilityResponse	1. The Test System sends a ChangeAvailabilityRequest with operationalStatus <i>Operative</i> and evse.id <Configured evseld> and evse.connectorId <Configured connectorId>
3. The Charging Station notifies the CSMS about the current state of the connectors.	4. The Test System responds accordingly.

Tool validations
* Step 2: Message ChangeAvailabilityResponse - status <i>Accepted</i> * Step 3: Message: StatusNotificationRequest - connectorStatus <i>Available</i> - evseld <Configured evseld> - connectorId <Configured connectorId> Message: NotifyEventRequest - eventData[0].trigger <i>Delta</i> - eventData[0].actualValue <i>"Available"</i> - eventData[0].component.name <i>"Connector"</i> - eventData[0].component.evse.id <Configured evseld> - eventData[0].component.evse.connectorId <Configured connectorId> - eventData[0].variable.name <i>"AvailabilityState"</i>
Post scenario validations: - A message to report the state of the connector has been received.

TC_G_09_CS: Change Availability EVSE - Operative to operative

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a ChangeAvailabilityResponse	1. The Test System sends a ChangeAvailabilityRequest with operationalStatus <i>Operative</i> and evse.id <i><Configured evseld></i>

Tool validations
* Step 2: Message ChangeAvailabilityResponse - status <i>Accepted</i>
Post scenario validations: N/a

TC_G_10_CS: Change Availability EVSE - Inoperative to inoperative

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
Reusable State: State is <i>Unavailable</i> for <Configured evseId>

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a ChangeAvailabilityResponse	1. The Test System sends a ChangeAvailabilityRequest with operationalStatus <i>Inoperative</i> and evse.id <Configured evseId>

Tool validations
* Step 2: Message ChangeAvailabilityResponse - status <i>Accepted</i>
Post scenario validations: - A message to report the state of a connector has been received for all connectors.

TC_G_11_CS: Change Availability EVSE - With ongoing transaction

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
Reusable State: State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a ChangeAvailabilityResponse	1. The Test System sends a ChangeAvailabilityRequest with operationalStatus <i>Inoperative</i> and evse.id <i><Configured evseld></i>
<u>Note(s)</u> : Wait for <i><Configured Transaction Duration></i>	
3. Execute Reusable State <i>StopAuthorized</i>	
4. The Charging Station notifies the CSMS about the current state of the connectors of the configured evse.	5. The Test System responds accordingly.
6. Execute Reusable State <i>EVConnectedPostSession</i>	
7. The Charging Station notifies the CSMS about the current state of the connectors of the configured evse.	8. The Test System responds accordingly.
9. Execute Reusable State <i>EVDisconnected</i>	
10. The Charging Station notifies the CSMS about the current state of the connectors of the configured evse.	11. The Test System responds accordingly.
12. Execute Reusable State <i>ParkingBayUnoccupied</i>	
13. The Charging Station notifies the CSMS about the current state of the connectors of the configured evse.	14. The Test System responds accordingly.
<u>Note(s)</u> : Steps 4, 5, 7, 8, 10, 11, 13, and 14 will only be executed if the previous step ended the transaction	

Tool validations

* Step 2:

Message **ChangeAvailabilityResponse**

- **status** *Scheduled*

* Step 4, 7, 10, 13:

Message: **StatusNotificationRequest**

- **connectorStatus** *Unavailable*

- **evseld** *<Configured evseld>*

- **connectorId** *<Configured connectorId>*

- when applicable this is also sent for other connectors of the same EVSE

Message: **NotifyEventRequest**

- **eventData[0].trigger** *Delta*

- **eventData[0].actualValue** *"Unavailable"*

- **eventData[0].component.name** *"Connector"*

- **eventData[0].variable.name** *"AvailabilityState"*

- **eventData[0].component.evse.id** *<Configured evseld>*

- **eventData[0].component.evse.connectorId** *<Configured connectorId>*

- when applicable this is also sent for other connectors of the same EVSE

Post scenario validations:

- A message to report the state of a connector has been received for all connectors.

TC_G_12_CS: Change Availability Charging Station - Operative to operative

Test case name	Change Availability Charging Station - Operative to operative
Test case Id	TC_G_12_CS
Use case Id(s)	G04
Requirement(s)	G04.FR.01, G04.FR.02, G04.FR.03, G04.FR.04, G04.FR.05
System under test	Charging Station
Description	<p>This test case describes how the CSMS requests the Charging Station to change the availability from inoperative to operative.</p> <p>A Charging Station is considered Operative when it is charging or ready for charging.</p> <p>A Charging Station is considered Inoperative when it does not allow any charging.</p>
Purpose	To verify if the Charging Station is able to perform the change availability from operative to operative according to the mechanism as described at the OCPP specification.
Prerequisite(s)	n/a

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a ChangeAvailabilityResponse	1. The Test System sends a ChangeAvailabilityRequest with operationalStatus <i>Operative</i>

Tool validations
* Step 2: Message ChangeAvailabilityResponse - status <i>Accepted</i>
Post scenario validations: - A message to report the state of a connector has been received for all connectors.

TC_G_13_CS: Change Availability Charging Station - Inoperative to inoperative

Test case name	Change Availability Charging Station - Inoperative to inoperative
Test case Id	TC_G_13_CS
Use case Id(s)	G04
Requirement(s)	G04.FR.01, G04.FR.02, G04.FR.03, G04.FR.04, G04.FR.05
System under test	Charging Station
Description	<p>This test case describes how the CSMS requests the Charging Station to change the availability from operative to inoperative.</p> <p>A Charging Station is considered Operative when it is charging or ready for charging.</p> <p>A Charging Station is considered Inoperative when it does not allow any charging.</p>
Purpose	To verify if the Charging Station is able to perform the change availability from Inoperative to Inoperative according to the mechanism as described at the OCPP specification.
Prerequisite(s)	n/a

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State: State is <i>Unavailable</i>

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a ChangeAvailabilityResponse	1. The Test System sends a ChangeAvailabilityRequest with operationalStatus <i>Inoperative</i>
3. The Charging Station notifies the CSMS about the current state of all connectors.	4. The Test System responds accordingly.

Tool validations
<p>* Step 2: Message ChangeAvailabilityResponse - status <i>Accepted</i></p> <p>* Step 3: Message: StatusNotificationRequest - connectorStatus <i>Unavailable</i></p> <p>Message: NotifyEventRequest - eventData[0].trigger <i>Delta</i> - eventData[0].actualValue <i>"Unavailable"</i> - eventData[0].component.name <i>"ChargingStation"</i> - eventData[0].variable.name <i>"AvailabilityState"</i></p>
Post scenario validations: - A message to report the state of a connector has been received for all connectors.

TC_G_14_CS: Change Availability Charging Station - With ongoing transaction

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

Reusable State:

State is *EnergyTransferStarted*

Main (Test scenario)

Charging Station	CSMS
2. The Charging Station responds with a ChangeAvailabilityResponse	1. The Test System sends a ChangeAvailabilityRequest with operationalStatus <i>Inoperative</i>
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 Test System responds accordingly.
<u>Note(s)</u> : Wait for <Configured Transaction Duration>	
5. Execute Reusable State <i>StopAuthorized</i>	
6. The Charging Station notifies the CSMS about the current state of the connectors of the configured evse.	7. The Test System responds accordingly.
8. Execute Reusable State <i>EVConnectedPostSession</i>	
9. The Charging Station notifies the CSMS about the current state of the connectors of the configured evse.	10. The Test System responds accordingly.
11. Execute Reusable State <i>EVDisconnected</i>	
12. The Charging Station notifies the CSMS about the current state of the connectors of the configured evse.	13. The Test System responds accordingly.
14. Execute Reusable State <i>ParkingBayUnoccupied</i>	
15. The Charging Station notifies the CSMS about the current state of the connectors of the configured evse.	16. The Test System responds accordingly.
<u>Note(s)</u> : Steps 6, 7, 9, 10, 12, 13, 15, and 16 will only be executed if the previous step ended the transaction	

Tool validations
<p>* Step 2:</p> <p>Message ChangeAvailabilityResponse</p> <ul style="list-style-type: none">- status <i>Scheduled</i> <p>* Step 7, 10, 13, 16:</p> <p>Message: StatusNotificationRequest</p> <ul style="list-style-type: none">- connectorStatus <i>Unavailable</i>- evseld not <i>0</i>- connectorId not <i>0</i> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none">- eventData[0].trigger <i>Delta</i>- eventData[0].actualValue <i>"Unavailable"</i>- eventData[0].component.name <i>"Connector"</i>- eventData[0].variable.name <i>"AvailabilityState"</i>
<p>Post scenario validations:</p> <ul style="list-style-type: none">- A message to report the state of a connector has been received for all connectors.

TC_G_15_CS: Change Availability Connector - Operative to operative

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a ChangeAvailabilityResponse	1. The Test System sends a ChangeAvailabilityRequest with operationalStatus <i>Operative</i> and evse.id <i><Configured evseld></i> and evse.connectorId <i><Configured connectorId></i>

Tool validations
* Step 2: Message ChangeAvailabilityResponse - status <i>Accepted</i>
Post scenario validations: - A message to report the state of a connector has been received for all connectors.

TC_G_16_CS: Change Availability Connector - Inoperative to inoperative

Test case name	Change Availability Connector - Inoperative to inoperative
Test case Id	TC_G_16_CS
Use case Id(s)	G03
Requirement(s)	G03.FR.01, G03.FR.02, G03.FR.03, G03.FR.04
System under test	Charging Station
Description	This test case covers how the CSMS requests the Charging Station to change the availability of one of the EVSEs from Operative to Inoperative. An EVSE is considered Operative in any status other than Faulted and Unavailable.
Purpose	To verify if the Charging Station is able to perform the change availability from inoperative to inoperative on one connector according to the mechanism as described at the OCPP specification.
Prerequisite(s)	n/a

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State: State is <i>Unavailable</i>

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a ChangeAvailabilityResponse	1. The Test System sends a ChangeAvailabilityRequest with operationalStatus <i>Inoperative</i> and evse.id <i><Configured evseId></i> and evse.connectorId <i><Configured connectorId></i>

Tool validations
* Step 2: Message ChangeAvailabilityResponse - status <i>Accepted</i>
Post scenario validations: - A message to report the state of a connector has been received for all connectors.

TC_G_17_CS: Change Availability Connector - With ongoing transaction

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
Reusable State: State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a ChangeAvailabilityResponse	1. The Test System sends a ChangeAvailabilityRequest with operationalStatus <i>Inoperative</i> and evse.id <i><Configured evseld></i> and evse.connectorId <i><Configured connectorId></i>
<u>Note(s)</u> : Wait for <i><Configured Transaction Duration></i>	
3. Execute Reusable State <i>StopAuthorized</i>	
4. The Charging Station notifies the CSMS about the current state of the connectors of the configured evse.	5. The Test System responds accordingly.
6. Execute Reusable State <i>EVConnectedPostSession</i>	
7. The Charging Station notifies the CSMS about the current state of the connectors of the configured evse.	8. The Test System responds accordingly.
9. Execute Reusable State <i>EVDisconnected</i>	
10. The Charging Station notifies the CSMS about the current state of the connectors of the configured evse.	11. The Test System responds accordingly.
12. Execute Reusable State <i>ParkingBayUnoccupied</i>	
13. The Charging Station notifies the CSMS about the current state of the connectors of the configured evse.	14. The Test System responds accordingly.
<u>Note(s)</u> : Steps 4, 5, 7, 8, 10, 11, 13, and 14 will only be executed if the previous step ended the transaction	

Tool validations

* Step 2:

Message **ChangeAvailabilityResponse**

- **status** *Scheduled*

* Step 7:

Message: **StatusNotificationRequest**

- **connectorStatus** *Unavailable*

- **evseld** *<Configured evseld>*

- **connectorId** *<Configured connectorId>*

- when applicable this is also sent for other connectors of the same EVSE

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 evseld>*

- **eventData[0].component.evse.connectorId** *<Configured connectorId>*

- **eventData[0].variable.name** *"AvailabilityState"*

- when applicable this is also sent for other connectors of the same EVSE

Post scenario validations:

- A message to report the state of a connector has been received for all connectors.

TC_G_18_CS: Change Availability EVSE - state persists across reboot

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
Reusable State: state is <i>Unavailable</i>

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a ChangeAvailabilityResponse	1. The Test System sends a ChangeAvailabilityRequest with operationalStatus <i>Inoperative</i> AND evse.id <Configured evseld>
3. The Charging Station notifies the CSMS about the current state of all connectors.	4. The Test System responds accordingly.
5. Execute Reusable State <i>Booted</i> <u>Note(s):</u> - After booting the charging station should send the following status: Message: StatusNotificationRequest - connectorStatus <i>Unavailable</i> - evseld <Configured evseld> Message: NotifyEventRequest - eventData[0].trigger <i>Delta</i> - eventData[0].actualValue "Unavailable" - eventData[0].component.name "Connector" - eventData[0].component.evse.id <Configured evseld> - eventData[0].variable.name "AvailabilityState"	

Tool validations

* Step 2:

Message **ChangeAvailabilityResponse**

- **status** *Accepted*

* Step 3:

Message: **StatusNotificationRequest**

- **evseld** not *0*

- **connectorId** not *0*

- **connectorStatus** *Unavailable* for **evseld** *<Configured evseld>*

- **connectorStatus** *Available* for **evseld** not *<Configured evseld>*

Message: **NotifyEventRequest**

- **eventData[0].actualValue** *Unavailable* for **evseld** *<Configured evseld>*

- **eventData[0].actualValue** *Available* for **evseld** not *<Configured evseld>*

Post scenario validations:

- A message to report the state of a connector has been received for all connectors.

TC_G_19_CS: Change Availability Connector - state persists across reboot

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
Reusable State: state is <i>Unavailable</i>

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>Booting</i>	
2. The Charging Station sends a BootNotificationRequest	3. The Test System responds with a BootNotificationResponse .
4. The Charging Station reports the status of all its connectors.	5. The Test System responds accordingly.
6. The Charging Station sends a SecurityEventNotificationRequest	7. The Test System responds with a SecurityEventNotificationResponse

Tool validations
<p>* Step 4:</p> <p>Message: StatusNotificationRequest</p> <ul style="list-style-type: none"> - evseld not 0 - connectorId not 0 - connectorStatus <i>Unavailable</i> for evseld <Configured evseld> and for connectorId <Configured ConnectorId> - connectorStatus <i>Available</i> for evseld not <Configured evseld> and for connectorId <Configured ConnectorId> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none"> - eventData[0].actualValue <i>Unavailable</i> for evseld <Configured evseld> and for connectorId <Configured ConnectorId> - eventData[0].actualValue <i>Available</i> for evseld not <Configured evseld> and for connectorId <Configured ConnectorId> <p>* Step 6:</p> <p>Message: SecurityEventNotificationRequest</p> <ul style="list-style-type: none"> - type "StartupOfTheDevice" or type "ResetOrReboot"
<p>Post scenario validations:</p> <ul style="list-style-type: none"> - A message to report the state of a connector has been received for all connectors.

TC_G_20_CS: Connector status Notification - Lock Failure

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)	<ul style="list-style-type: none"> - 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
Reusable State: State is <i>EVConnectedPreSession</i> <u>Note(s):</u> - 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 Test System responds with a NotifyEventResponse

Tool validations
* Step 2: Message: NotifyEventRequest - eventData[0].trigger <i>Delta</i> - eventData[0].component.name <i>"ConnectorPlugRetentionLock"</i> - eventData[0].variable.name <i>"Problem"</i> - eventData[0].actualValue <i>"true"</i>
Post scenario validations: - The charging station did not start charging

TC_G_21_CS: Change Availability Charging Station - state persists across reboot

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
Reusable State: State is <i>Unavailable</i>

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>Booting</i>	
2. The Charging Station sends a BootNotificationRequest .	3. The Test System responds with a BootNotificationResponse .
4. The Charging Station reports the status of all its connectors.	5. The Test System responds accordingly.
6. The Charging Station sends a SecurityEventNotificationRequest	7. The Test System responds with a SecurityEventNotificationResponse

Tool validations
<p>* Step 4:</p> <p>Message: StatusNotificationRequest</p> <ul style="list-style-type: none"> - evseld not 0 - connectorId not 0 - connectorStatus <i>Unavailable</i> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none"> - eventData[0].trigger <i>Delta</i> - eventData[0].actualValue <i>"Unavailable"</i> - eventData[0].variable.name <i>"AvailabilityState"</i> <p>* Step 6:</p> <p>Message: SecurityEventNotificationRequest</p> <ul style="list-style-type: none"> - type <i>"StartupOfTheDevice"</i> or type <i>"ResetOrReboot"</i> <p>Post scenario validations:</p> <ul style="list-style-type: none"> - A message to report the state of a connector has been received for all connectors.

H Reservation

TC_H_01_CS: Reserve a specific EVSE - Accepted - Valid idToken

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 evseld.
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 <i>true</i>

Before (Preparations)
Configuration State: ReservationCtrlr.ReservationEnabled is <i>true</i> (If implemented)
Memory State: N/a
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>Reserved</i> for <Configured evseld>	
2. Execute Reusable State <i>Authorized</i>	
<u>Note(s)</u> : - <Configured valid idToken fields> are used for the authorization.	
3. Execute Reusable State <i>EnergyTransferStarted</i>	

Tool validations
<p>* Step 2:</p> <p>After authorization, connector status must change from Reserved to Available</p> <p>Message: StatusNotificationRequest</p> <ul style="list-style-type: none"> - evseld <configured evseld> - connectorId <configured connectorId> - connectorStatus must be Available <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none"> - 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 evseld> - connector.id <configured connectorId> <p>Post scenario validations: N/a</p>

TC_H_02_CS: Reserve a specific EVSE - Accepted - Different idToken

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 evseld. Starting a transaction can be done in two ways (this is configurable by the Test System; 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 <i>true</i>

Before (Preparations)

Configuration State: ReservationCtrlr.ReservationEnabled is <i>true</i> (If implemented)
Memory State: N/a
Reusable State(s): State is <Configured evseld> is <i>Reserved</i> for <Configured valid_idtoken1_idtoken> State is <i>EVConnectedPreSession</i>

Main A (Test scenario)

Charging Station	CSMS
Manual action: Authorize with <Configured valid_idtoken2_idtoken>.	
Execute reusable state <i>Authorized</i>	
Note(s): The test is a PASS, if the Test System 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 Test System sends a RequestStartTransactionRequest with idToken.idToken <Configured valid_idtoken2_idtoken> idToken.type <Configured valid_idtoken2_type> evseld <Configured evseld>

Tool validations

* Step 2:
Message: **RequestStartTransactionResponse**
- **status** must be *Rejected*

Post scenario validations:
N/a

TC_H_03_CS: Reserve a specific EVSE - Occupied - EVSE Reserved

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 evseld.
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 <i>true</i>

Before (Preparations)
Configuration State: ReservationCtrlr.ReservationEnabled is <i>true</i> (If implemented)
Memory State: <Configured evseld> is <i>Reserved</i>
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a ReserveNowResponse	1. The Test System sends a ReserveNowRequest with evseld is <Configured evseld> idToken.idToken <Configured valid_idtoken2_idtoken> idToken.type <Configured valid_idtoken2_type>

Tool validations
* Step 2: Message: ReserveNowResponse - status must be <i>Occupied</i>
Post scenario validations: N/a

TC_H_04_CS: Reserve a specific EVSE - Occupied - EVSE Occupied

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 evseld.
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 <i>true</i>

Before (Preparations)
Configuration State: ReservationCtrlr.ReservationEnabled is <i>true</i> (If implemented)
Memory State: State is <i>EnergyTransferStarted</i>
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a ReserveNowResponse	1. The Test System sends a ReserveNowRequest with evseld is <i><Configured evseld></i> idToken.idToken <i><Configured valid_idtoken2_idtoken></i> idToken.type <i><Configured valid_idtoken2_type></i>

Tool validations
* Step 2: Message: ReserveNowResponse - status must be <i>Occupied</i>
Post scenario validations: N/a

TC_H_06_CS: Reserve a specific EVSE - Unavailable

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 evseld.
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 <i>true</i>

Before (Preparations)
Configuration State: ReservationCtrlr.ReservationEnabled is <i>true</i> (If implemented)
Memory State: <Configured evseld> is <i>Unavailable</i>
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a ReserveNowResponse	1. The Test System sends a ReserveNowRequest with evseld is <Configured evseld> idToken.idToken <Configured valid_idtoken2_idtoken> idToken.type <Configured valid_idtoken2_type>

Tool validations
* Step 2: Message: ReserveNowResponse - status must be <i>Unavailable</i>
Post scenario validations: N/a

TC_H_07_CS: Reserve a specific EVSE - Reservation Ended / not used

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 evseld.
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 <i>true</i>

Before (Preparations)

Configuration State:

ReservationCtrlr.ReservationEnabled is *true* (If implemented)

Memory State:

<Configured evseld> 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. <u>Note(s):</u> - The Test System expects that the Charging Station sets the <i>availabilityState</i> of the EVSE and corresponding connectors back to <i>Available</i> after the expiry time of 60 seconds is reached.	2. The Test System responds accordingly.
3. The Charging Station sends a ReservationStatusUpdateRequest .	4. The Test System responds with a ReservationStatusUpdateResponse .
5. Execute Reusable State <i>Authorized</i> <u>Note(s):</u> - <Configured valid idtoken fields2> are used for the authorization.	
6. Execute Reusable State <i>EnergyTransferStarted</i>	

Tool validations
<p>* Step 1:</p> <p>Message: StatusNotificationRequest</p> <ul style="list-style-type: none">- connectorStatus must be <i>Available</i> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none">- eventData[0].trigger must be <i>Delta</i>- eventData[0].actualValue must be <i>Available</i>- eventData[0].component.name must be <i>Connector</i>- eventData[0].variable.name must be <i>AvailabilityState</i> <p>(Reporting the <i>AvailabilityState</i> of the EVSE component itself is optional.)</p> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none">- eventData[0].trigger must be <i>Delta</i>- eventData[0].actualValue must be <i>Available</i>- eventData[0].component.name must be <i>EVSE</i>- eventData[0].variable.name must be <i>AvailabilityState</i> <p>* Step 3:</p> <p>Message: ReservationStatusUpdateRequest</p> <ul style="list-style-type: none">- reservationId must be <i><Generated reservationId></i>- reservationUpdateStatus must be <i>Expired</i>
<p>Post scenario validations:</p> <p>N/a</p>

TC_H_08_CS: Reserve an unspecified EVSE - Accepted

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 evseld.
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)	<ul style="list-style-type: none"> - The configuration variable ReservationCtrlr.ReservationAvailable is implemented with value <i>true</i> - The Charging Station has the configuration variable ReservationNonEvseSpecific implemented with value <i>true</i>

Before (Preparations)
Configuration State: ReservationCtrlr.ReservationEnabled is <i>true</i> (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 Test System sends a ReserveNowRequest with evseld 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. <u>Note(s):</u> - If the Charging Station has only one EVSE, it sets the <i>availabilityState</i> of the EVSE and corresponding connectors to <i>Reserved</i> . - Reporting the <i>AvailabilityState</i> of the EVSE component itself is optional.	4. The Test System responds accordingly.
5. Execute Reusable State <i>Authorized</i> <u>Note(s):</u> - <Configured valid_idtoken_idtoken> is used for the authorization.	
6. Execute Reusable State <i>EnergyTransferStarted</i>	

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

TC_H_09_CS: Reserve an unspecified EVSE - Occupied - EVSE Reserved

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 evseld.
Purpose	To verify if the Charging Station is able to respond with status Occupied, when all EVSE are already reserved.
Prerequisite(s)	<ul style="list-style-type: none"> - The configuration variable ReservationCtrlr.ReservationAvailable is implemented with value <i>true</i> - The Charging Station has the configuration variable ReservationNonEvseSpecific implemented with value <i>true</i>

Before (Preparations)
Configuration State: ReservationCtrlr.ReservationEnabled is <i>true</i> (If implemented)
Memory State: N/a
Reusable State(s): All EVSE are <i>Reserved</i>

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a ReserveNowResponse	1. The Test System sends a ReserveNowRequest with evseld is Omitted idToken.idToken <Configured valid_idtoken_idtoken> idToken.type <Configured valid_idtoken_type>

Tool validations
* Step 2: Message: ReserveNowResponse - status must be <i>Occupied</i>
Post scenario validations: N/a

TC_H_10_CS: Reserve an unspecified EVSE - Occupied - EVSE Occupied

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 evseld.
Purpose	To verify if the Charging Station is able to respond with status Occupied, when all EVSE are occupied.
Prerequisite(s)	<ul style="list-style-type: none"> - The configuration variable ReservationCtrlr.ReservationAvailable is implemented with value <i>true</i> - The Charging Station has the configuration variable ReservationNonEvseSpecific implemented with value <i>true</i>

Before (Preparations)
Configuration State: ReservationCtrlr.ReservationEnabled is <i>true</i> (If implemented)
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferStarted</i> for all EVSE

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a ReserveNowResponse	1. The Test System sends a ReserveNowRequest with evseld 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 <i>Occupied</i>
Post scenario validations: N/a

TC_H_12_CS: Reserve an unspecified EVSE - Unavailable

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 evseld.
Purpose	To verify if the Charging Station is able to respond with status Unavailable, when all EVSE are unavailable.
Prerequisite(s)	<ul style="list-style-type: none"> - The configuration variable ReservationCtrlr.ReservationAvailable is implemented with value <i>true</i> - The Charging Station has the configuration variable ReservationNonEvseSpecific implemented with value <i>true</i>

Before (Preparations)
Configuration State: ReservationCtrlr.ReservationEnabled is <i>true</i> (If implemented)
Memory State: Charging Station is <i>Unavailable</i>
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a ReserveNowResponse	1. The Test System sends a ReserveNowRequest with evseld is Omitted idToken.idToken <Configured valid_idtoken_idtoken> idToken.type <Configured valid_idtoken_type>

Tool validations
* Step 2: Message: ReserveNowResponse - status must be <i>Unavailable</i>
Post scenario validations: N/a

TC_H_13_CS: Reserve an unspecified EVSE - Rejected

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 evseld.
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)	<ul style="list-style-type: none"> - The configuration variable ReservationCtrlr.ReservationAvailable is implemented with value <i>true</i> - The Charging Station does NOT have the configuration variable ReservationNonEvseSpecific implemented OR the Charging Station does have it implemented with value <i>false</i>

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 Test System sends a ReserveNowRequest with evseld 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

TC_H_14_CS: Reserve an unspecified EVSE - Amount of EVSEs available equals the amount of reservations

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 evseld.
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)	<ul style="list-style-type: none"> - The configuration variable ReservationCtrlr.ReservationAvailable is implemented with value <i>true</i> - The Charging Station has the configuration variable ReservationNonEvseSpecific implemented with value <i>true</i>

Before (Preparations)
Configuration State: ReservationCtrlr.ReservationEnabled is <i>true</i> (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	<p>1. The Test System sends a ReserveNowRequest with evseld is Omitted idToken is <A different idToken for every reservation being made></p> <p><u>Note(s):</u> - This step will be executed the amount of times equal to the amount of EVSE the Charging Station has.</p>
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 Test System responds accordingly.

Tool validations
<p>* Step 2:</p> <p>Message: ReserveNowResponse</p> <ul style="list-style-type: none"> - status must be <i>Accepted</i> <p>* Step 3:</p> <p>Message: StatusNotificationRequest</p> <ul style="list-style-type: none"> - connectorStatus must be <i>Reserved</i> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none"> - eventData[0].trigger must be <i>Delta</i> - eventData[0].actualValue must be <i>Reserved</i> - eventData[0].component.name must be <i>Connector</i> - eventData[0].variable.name must be <i>AvailabilityState</i> <p>(Optional)</p> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none"> - eventData[0].trigger must be <i>Delta</i> - eventData[0].actualValue must be <i>Reserved</i> - eventData[0].component.name must be <i>EVSE</i> - eventData[0].variable.name must be <i>AvailabilityState</i>

Tool validations
Post scenario validations: N/a

TC_H_15_CS: Reserve a connector with a specific type - Success

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)	<ul style="list-style-type: none"> - The configuration variable ReservationCtrlr.ReservationAvailable is implemented with value <i>true</i> - The Charging Station supports the reservation of a specific connector type, that is part of the ConnectorEnumType.

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 Test System 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. <u>Note(s):</u> - 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 Test System responds accordingly.
5. Execute Reusable State <i>Authorized</i> <u>Note(s):</u> - <Configured valid idToken fields> are used for the authorization.	
6. Execute Reusable State <i>EnergyTransferStarted</i>	

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

TC_H_16_CS: Reserve a connector with a specific type - Amount of available connectors of a type equals the amount of reservations

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 <i>true</i> - The Charging Station supports the reservation of a specific connector type, that is part of the ConnectorEnumType.

Before (Preparations)
Configuration State: ReservationCtrlr.ReservationEnabled is <i>true</i> (If implemented) <i>All EVSEs should be reserved</i>
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 Test System 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 <i>Occupied</i>
Post scenario validations: N/a

TC_H_17_CS: Cancel reservation of an EVSE - Success

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 <i>true</i>

Before (Preparations)
Configuration State: ReservationCtrlr.ReservationEnabled is <i>true</i> (If implemented)
Memory State: <Configured evseId> is <i>Reserved</i>
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a CancelReservationResponse	1. The Test System sends a CancelReservationRequest with reservationId is <Generated reservationId>
3. The Charging Station notifies the CSMS about the status change of the connector.	4. The Test System responds accordingly.

Tool validations
<p>* Step 2:</p> <p>Message: CancelReservationResponse</p> <ul style="list-style-type: none"> - status must be <i>Accepted</i> <p>* Step 3:</p> <p>For each connector on the <Configured evseId> one of the following messages must be sent:</p> <p>Message: StatusNotificationRequest</p> <ul style="list-style-type: none"> - connectorStatus must be <i>Available</i> - evseId must be <Configured evseId> - connectorId must be <Configured connectorId> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none"> - trigger must be <i>Delta</i> - actualValue must be <i>"Available"</i> - component.name must be <i>"Connector"</i> - evse.id must be <Configured evseId> - eves.connectorId must be <Configured connectorId> - variable.name must be <i>"AvailabilityState"</i> <p>Post scenario validations:</p> <p>N/a</p>

TC_H_18_CS: Cancel reservation of an EVSE - Rejected

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 <i>true</i>

Before (Preparations)
Configuration State: ReservationCtrlr.ReservationEnabled is <i>true</i> (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 Test System sends a CancelReservationRequest with reservationId is 1

Tool validations
* Step 2: Message: CancelReservationResponse - status must be <i>Rejected</i>
Post scenario validations: N/a

TC_H_19_CS: Reserve a specific EVSE - Use a reserved EVSE with GroupId

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 <i>true</i>

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 Test System sends a ReserveNowRequest with evseld is <Configured evseld> 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. <u>Note(s):</u> - The Test System expects that the Charging Station sets the <i>availabilityState</i> of the EVSE and corresponding connectors to <i>Reserved</i> . - Reporting the <i>AvailabilityState</i> of the EVSE component itself is optional.	4. The Test System responds accordingly.
3. Execute Reusable State <i>Authorized</i> <u>Note(s):</u> - <Configured valid_idtoken_idtoken2> with <Configured groupIdToken> is used for the authorization.	
4. Execute Reusable State <i>EnergyTransferStarted</i>	

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

TC_H_21_CS: Charging Station cancels reservation when Unavailable

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 <i>Inoperative</i> .
Prerequisite(s)	The configuration variable ReservationCtrlr.ReservationAvailable is implemented with value <i>true</i>

Before (Preparations)
Configuration State: ReservationCtrlr.ReservationEnabled is <i>true</i> (If implemented)
Memory State: <Configured evseId> is <i>Reserved</i>
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a ChangeAvailabilityResponse	1. The Test System sends a ChangeAvailabilityRequest with operationalStatus <i>Inoperative</i> and evse.id <Configured evseId>
3. The Charging Station notifies the CSMS about the status change of the connector. <u>Note(s):</u> - This step needs to be executed for all connectors of the specified EVSE. - Reporting the AvailabilityState of the EVSE itself is optional.	4. The Test System responds accordingly.
5. The Charging Station sends a ReservationStatusUpdateRequest .	6. The Test System responds with a ReservationStatusUpdateResponse .
8. The Charging Station responds with a ChangeAvailabilityResponse	7. The Test System sends a ChangeAvailabilityRequest with operationalStatus <i>Operative</i> and evse.id <Configured evseId>
9. The Charging Station notifies the CSMS about the status change of the connector. <u>Note(s):</u> - This step needs to be executed for all connectors of the specified EVSE. - Reporting the AvailabilityState of the EVSE itself is optional.	10. The Test System responds accordingly.
11. Execute Reusable State <i>Authorized</i> <u>Note(s):</u> - <Configured valid idtoken fields2> are used for the authorization.	
12. Execute Reusable State <i>EnergyTransferStarted</i>	

Tool validations
<p>* Step 2:</p> <p>Message: ChangeAvailabilityResponse</p> <ul style="list-style-type: none"> - status <i>Accepted</i> <p>* Step 3:</p> <p>Message: StatusNotificationRequest</p> <ul style="list-style-type: none"> - connectorStatus must be <i>Unavailable</i> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none"> - eventData[0].trigger must be <i>Delta</i> - eventData[0].actualValue must be <i>Unavailable</i> - eventData[0].component.name must be <i>Connector</i> - eventData[0].variable.name must be <i>AvailabilityState</i> <p>(Reporting the AvailabilityState of the EVSE component itself is optional.)</p> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none"> - eventData[0].trigger must be <i>Delta</i> - eventData[0].actualValue must be <i>Unavailable</i> - eventData[0].component.name must be <i>EVSE</i> - eventData[0].variable.name must be <i>AvailabilityState</i> <p>* Step 5:</p> <p>Message: ReservationStatusUpdateRequest</p> <ul style="list-style-type: none"> - reservationId must be <i><Generated reservationId></i> - reservationUpdateStatus must be <i>Removed</i> <p>* Step 8:</p> <p>Message: ChangeAvailabilityResponse</p> <ul style="list-style-type: none"> - status <i>Accepted</i> <p>* Step 9:</p> <p>Message: StatusNotificationRequest</p> <ul style="list-style-type: none"> - connectorStatus must be <i>Available</i> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none"> - eventData[0].trigger must be <i>Delta</i> - eventData[0].actualValue must be <i>Available</i> - eventData[0].component.name must be <i>Connector</i> - eventData[0].variable.name must be <i>AvailabilityState</i> <p>(Reporting the AvailabilityState of the EVSE component itself is optional.)</p> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none"> - eventData[0].trigger must be <i>Delta</i> - eventData[0].actualValue must be <i>Available</i> - eventData[0].component.name must be <i>EVSE</i> - eventData[0].variable.name must be <i>AvailabilityState</i>
<p>Post scenario validations:</p> <p>N/a</p>

TC_H_22_CS: Reserve a specific EVSE - Configured to Reject

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 evseld.
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 <i>false</i>

Before (Preparations)
Configuration State: ReservationCtrlr.ReservationEnabled is <i>false</i> (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 Test System sends a ReserveNowRequest

Tool validations
* Step 2: Message: ReserveNowResponse - status <i>Rejected</i>
Post scenario validations: N/a

TC_H_23_CS: Reserve a specific EVSE - Replace reservation

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 evseld.
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 <i>true</i>

Before (Preparations)
Configuration State: ReservationCtrlr.ReservationEnabled is <i>true</i> (If implemented)
Memory State: A reservation is valid on <Configured evseld> 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 Test System sends a ReserveNowRequest with id <Configured reservationId> evseld is <Configured evseld> idToken.idToken <Configured valid_idtoken_idtoken2> idToken.type <Configured valid_idtoken_type2>
3. Execute Reusable State <i>Authorized</i>	
Note(s): - <Configured valid idToken2 fields> are used for the authorization.	
4. Execute Reusable State <i>EnergyTransferStarted</i>	

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

TC_H_24_CS: Reserve an unspecified EVSE - GroupIdToken

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 evseld.
Purpose	To verify if the Charging Station is able to reserve a unspecific EVSE, until the EV Driver with the specified groupIdToken arrives. Since all other EVSEs are reserved, this reservation will force the last EVSE to also become reserved.
Prerequisite(s)	The configuration variable ReservationCtrlr.ReservationAvailable is implemented with value <i>true</i>

Before (Preparations)
Configuration State: ReservationCtrlr.ReservationEnabled is <i>true</i> (If implemented)
Memory State: N/a
Reusable State(s): All EVSEs except one are <i>Reserved</i>

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a ReserveNowResponse	1. The Test System sends a ReserveNowRequest with idToken.idToken is <i><Configured valid_idtoken></i> groupIdToken.idToken is <i><Configured group_idtoken></i>
3. The Charging Station notifies the CSMS about the status change of the connector. <u>Note(s):</u> - The Test System expects that the Charging Station sets the <i>availabilityState</i> of the only not reserved EVSE and corresponding connectors to <i>Reserved</i> . - Reporting the <i>AvailabilityState</i> of the EVSE component itself is optional.	4. The Test System responds accordingly.
5. Execute Reusable State <i>Authorized</i> <u>Note(s)</u> : - <i><Configured valid_idtoken2></i> is used for the authorization.	
6. Execute Reusable State <i>EnergyTransferStarted</i>	

Tool validations
<p>* Step 2:</p> <p>Message: ReserveNowResponse</p> <ul style="list-style-type: none">- status must be <i>Accepted</i> <p>* Step 3:</p> <p>Message: StatusNotificationRequest</p> <ul style="list-style-type: none">- connectorStatus must be <i>Reserved</i> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none">- eventData[0].trigger must be <i>Delta</i>- eventData[0].actualValue must be <i>Reserved</i>- eventData[0].component.name must be <i>Connector</i>- eventData[0].variable.name must be <i>AvailabilityState</i> <p>(Optional)</p> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none">- eventData[0].trigger must be <i>Delta</i>- eventData[0].actualValue must be <i>Reserved</i>- eventData[0].component.name must be <i>EVSE</i>- eventData[0].variable.name must be <i>AvailabilityState</i>
<p>Post scenario validations:</p> <p>N/a</p>

I Tariff and Cost

TC_I_01_CS: Show EV Driver running total cost during charging - costUpdatedRequest

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
<u>Manual Action:</u> Present valid idToken	
1. The Charging Station sends an AuthorizeRequest	2. The Test System responds with an AuthorizeResponse with - idTokenInfo.status <i>Accepted</i> - 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 <i>ParkingBayOccupancy</i>	4. The Test System responds with a TransactionEventResponse with - idTokenInfo.status <i>Accepted</i>
5. Execute Reusable State <i>EnergyTransferStarted</i>	
6. The Charging Station sends an TransactionEventRequest	7. The Test System responds with an TransactionEventResponse with - updatedPersonalMessage.content <Configured Cost>
9. The Charging Station responds with a CostUpdatedResponse	8. The Test System sends a CostUpdatedRequest with - totalCost <Configured Cost2> - transactionId <Configured transactionId>
<u>Note(s):</u> Step 6, 7, 8, and 9 are repeated <i>n</i> times	

Tool validations
<div>* Step 1: Message AuthorizeRequest - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type> * Step 3: Message TransactionEventRequest - triggerReason <i>Authorized</i> - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type> if transaction was already started - eventType <i>Updated</i> else - eventType <i>Started</i></div>
<div>Post scenario validations: - N/a</div>

TC_I_02_CS: Show EV Driver Final Total Cost After Charging

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 <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action</u> : <i>Present valid idToken</i>	
1. Execute Reusable State <i>StopAuthorized</i> <u>Note</u> : IF Message TransactionEventRequest - eventType <i>Ended</i> THEN Message TransactionEventResponse - totalCost <i><Generated Cost></i>	
2. Execute Reusable State <i>EVConnectedPostSession</i> <u>Note</u> : IF Message TransactionEventRequest - eventType <i>Ended</i> THEN Message TransactionEventResponse - totalCost <i><Generated Cost></i>	
3. Execute Reusable State <i>EVDIsconnected</i> <u>Note</u> : IF Message TransactionEventRequest - eventType <i>Ended</i> THEN Message TransactionEventResponse - totalCost <i><Generated Cost></i>	
4. Execute Reusable State <i>ParkingBayUnoccupied</i> <u>Note</u> : IF Message TransactionEventRequest - eventType <i>Ended</i> THEN Message TransactionEventResponse - totalCost <i><Generated Cost></i>	

Tool validations
N/a
Post scenario validations: - N/a

TC_I_07_CS: Show EV Driver running total cost during charging - transactionEventResponse

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 <i>EVConnectedPreSession</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action:</u> Present valid idToken	
1. The Charging Station sends an AuthorizeRequest	2. The Test System responds with an AuthorizeResponse with - idTokenInfo.status <i>Accepted</i> - idTokenInfo.personalMessage.content <i><Configured Cost></i>
3. The Charging Station sends a TransactionEventRequest <u>Note(s):</u> - This step needs to be executed when TxStartPoint contains <i>Authorized</i> OR the transaction already started. So in the case TxStartPoint contains <i>ParkingBayOccupancy</i>	4. The Test System responds with a TransactionEventResponse with - idTokenInfo.status <i>Accepted</i>
5. Execute Reusable State <i>EnergyTransferStarted</i>	
6. The Charging Station sends an TransactionEventRequest	7. The Test System responds with an TransactionEventResponse with - updatedPersonalMessage.content <i><Configured Cost></i>
9. The Charging Station responds with a TransactionEventResponse	8. The Test System sends a TransactionEventRequest with - totalCost <i><Configured Cost2></i> - transactionId <i><Configured transactionId></i>
<u>Note(s):</u> Step 6, 7, 8, and 9 are repeated <i>n</i> times	

Tool validations
<div>* Step 1: Message AuthorizeRequest - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type> * Step 3: Message TransactionEventRequest - triggerReason <i>Authorized</i> - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type> if transaction was already started - eventType <i>Updated</i> else - eventType <i>Started</i></div>
<div>Post scenario validations: - N/a</div>

TC_I_100_CS: Set Default Tariff - validFrom

Test case name	Set Default Tariff - validFrom
Test case Id	TC_I_100_CS
Use case Id(s)	I07, I09, I10
Requirement(s)	I07.FR.10, I07.FR.20, I07.FR.21, I07.FR.22, I07.FR.23, I09.FR.02, I09.FR.07, I10.FR.01
System under test	Charging Station
Description	To set the default tariff on the charging station, for example for ad hoc charging, or when there is no driver-specific tariff.
Purpose	To verify if the Charging Station correctly replaces tariffs based on the tariff's validFrom fields.
Prerequisite(s)	- The Charging Station supports local cost calculation using the Tariff mechanism (TariffCostCtrlr.Enabled[Tariff] exists <i>ReadWrite</i> or is <i>true</i> and <i>ReadOnly</i>).

Before (Preparations)

Configuration State:
- **TariffCostCtrlr.Enabled[Tariff]** is *true*

Memory State:
N/a

Reusable State:
N/a

Main (Test scenario)

Charging Station	CSMS
2. The Charging Station responds with SetDefaultTariffResponse	1. The Test System sends a SetDefaultTariffRequest with evseld 0 tariff.tariffId <i>Test System1A</i> tariff.currency <i>EUR</i> tariff.energy.prices[0].priceKwh <i>1.00</i>
4. The Charging Station responds with SetDefaultTariffResponse	3. The Test System sends a SetDefaultTariffRequest with evseld 0 tariff.tariffId <i>Test System1B</i> tariff.validFrom <i><Current DateTime - 1:00 hour></i> tariff.currency <i>EUR</i> tariff.energy.prices[0].priceKwh <i>2.00</i>
6. The Charging Station responds with GetTariffsResponse	5. The Test System sends a GetTariffsRequest with evseld 0
8. The Charging Station responds with ClearTariffsResponse	7. The Test System sends a ClearTariffsRequest
10. The Charging Station responds with SetDefaultTariffResponse	9. The Test System sends a SetDefaultTariffRequest with evseld 0 tariff.tariffId <i>Test System1A</i> tariff.validFrom <i><Current DateTime - 2:00 hour></i> tariff.currency <i>EUR</i> tariff.energy.prices[0].priceKwh <i>1.00</i>
12. The Charging Station responds with SetDefaultTariffResponse	11. The Test System sends a SetDefaultTariffRequest with evseld 0 tariff.tariffId <i>Test System1B</i> tariff.validFrom <i><Current DateTime - 1:00 hour></i> tariff.currency <i>EUR</i> tariff.energy.prices[0].priceKwh <i>2.00</i>
14. The Charging Station responds with GetTariffsResponse	13. The Test System sends a GetTariffsRequest with evseld 0

Main (Test scenario)	
16. The Charging Station responds with ClearTariffsResponse	15. The Test System sends a ClearTariffsRequest
18. The Charging Station responds with SetDefaultTariffResponse	17. The Test System sends a SetDefaultTariffRequest with evseld 0 tariff.tariffId Test System1A tariff.validFrom <Current DateTime - 0:00 hour> tariff.currency EUR tariff.energy.prices[0].priceKwh 1.00
20. The Charging Station responds with SetDefaultTariffResponse	19. The Test System sends a SetDefaultTariffRequest with evseld 0 tariff.tariffId Test System1B tariff.validFrom <Current DateTime - 1:00 hour> tariff.currency EUR tariff.energy.prices[0].priceKwh 2.00
22. The Charging Station responds with GetTariffsResponse	21. The Test System sends a GetTariffsRequest with evseld 0
24. The Charging Station responds with ClearTariffsResponse	23. The Test System sends a ClearTariffsRequest
26. The Charging Station responds with SetDefaultTariffResponse	25. The Test System sends a SetDefaultTariffRequest with evseld 0 tariff.tariffId Test System1A tariff.validFrom <omitted> tariff.currency EUR tariff.energy.prices[0].priceKwh 1.00
28. The Charging Station responds with SetDefaultTariffResponse	27. The Test System sends a SetDefaultTariffRequest with evseld 0 tariff.tariffId Test System1B tariff.validFrom <omitted> tariff.currency EUR tariff.energy.prices[0].priceKwh 2.00
30. The Charging Station responds with GetTariffsResponse	29. The Test System sends a GetTariffsRequest with evseld 0

Tool validations

* Step 2:
 Message **SetDefaultTariffResponse**
 - **status** must be *Accepted*

* Step 4:
 Message **SetDefaultTariffResponse**
 - **status** must be *Accepted*

* Step 6:
 Message **GetTariffsResponse**
 - **status** must be *Accepted*
 - **tariffAssignments[0].tariffId** must be *Test System1B*
 - **tariffAssignments[0].validFrom** must be *tariff.validFrom* sent in step 4

* Step 8:
 Message **ClearTariffsResponse**
 - **tariffIds[0].clearTariffsResult.tariffId** must be *Test System1B*
 - **tariffIds[0].clearTariffsResult.status** must be *Accepted*

* Step 10:
 Message **SetDefaultTariffResponse**
 - **status** must be *Accepted*

* Step 12:
 Message **SetDefaultTariffResponse**
 - **status** must be *Accepted*

* Step 14:
 Message **GetTariffsResponse**
 - **status** must be *Accepted*
 - **tariffAssignments[0].tariffId** must be *Test System1B*
 - **tariffAssignments[0].validFrom** must be *tariff.validFrom* sent in step 11

* Step 16:
 Message **ClearTariffsResponse**
 - **tariffIds[0].clearTariffsResult.tariffId** must be *Test System1B*
 - **tariffIds[0].clearTariffsResult.status** must be *Accepted*

* Step 18:
 Message **SetDefaultTariffResponse**
 - **status** must be *Accepted*

* Step 20:
 Message **SetDefaultTariffResponse**
 - **status** must be *Accepted*

* Step 22:
 Message **GetTariffsResponse**
 - **status** must be *Accepted*
 - **tariffAssignments[0].tariffId** must be *Test System1A*
 - **tariffAssignments[0].validFrom** must be *tariff.validFrom* sent in step 17

Tool validations
<div>* Step 24: Message ClearTariffsResponse<ul style="list-style-type: none">- tariffIds[0].clearTariffsResult.tariffId must be <i>Test System1A</i>- tariffIds[0].clearTariffsResult.status must be <i>Accepted</i></div> <div>* Step 26: Message SetDefaultTariffResponse<ul style="list-style-type: none">- status must be <i>Accepted</i></div> <div>* Step 28: Message SetDefaultTariffResponse<ul style="list-style-type: none">- status must be <i>Accepted</i></div> <div>* Step 30: Message GetTariffsResponse<ul style="list-style-type: none">- status must be <i>Accepted</i>- tariffAssignments[0].tariffId must be <i>Test System1B</i>- tariffAssignments[0].validFrom must be <i><omitted></i></div>
<div>Post scenario validations: N/a</div>

TC_I_101_CS: Set Default Tariff - startTimeOfDay, endTimeOfDay

Test case name	Set Default Tariff - startTimeOfDay, endTimeOfDay
Test case Id	TC_I_101_CS
Use case Id(s)	I07, I12
Requirement(s)	I07.FR.10, I07.FR.14, I12.FR.02, I12.FR.03, I12.FR.10, I12.FR.16
System under test	Charging Station
Description	To set the default tariff on the charging station, for example for ad hoc charging, or when there is no driver-specific tariff.
Purpose	To verify if the Charging Station supports the start/endTimeOfDay conditions
Prerequisite(s)	- The Charging Station supports local cost calculation using the Tariff mechanism (TariffCostCtrlr.Enabled[Tariff] exists and <i>ReadWrite</i> or is <i>true</i> and <i>ReadOnly</i>).

Before (Preparations)
Configuration State: - TariffCostCtrlr.Enabled[Tariff] is <i>true</i> - TariffCostCtrlr.Enabled[RunningCost] is <i>true</i> - TariffCostCtrlr.Enabled[Cost] is <i>true</i> - TariffCostCtrlr.Interval[Cost] is 15 - TariffCostCtrlr.ConditionsSupported[Tariff] is <i>true</i> - SampledDataCtrlr.TxUpdatedInterval = 0
Memory State: N/a
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with SetDefaultTariffResponse	1. The Test System sends a SetDefaultTariffRequest with evseld 0 tariff.tariffId <i>Test System1</i> tariff.currency <i>EUR</i> tariff.chargingTime.prices[0].priceMinute <i>0.00</i> tariff.chargingTime.prices[0].conditions.startTimeOfDay <Current Time> tariff.chargingTime.prices[0].conditions.endTimeOfDay <Current Time + 1 minute> tariff.chargingTime.prices[1].priceMinute <i>10.00</i>
3. Execute Reusable State <i>EnergyTransferStarted</i>	
4. The Charging Station sends a TransactionEventRequest	5. The Test System responds with a TransactionEventResponse
<u>Note:</u> Steps 4 and 5 will repeat every 15 seconds for 1 minute using price of 0.00 EUR/minute	
6. The Charging Station sends a TransactionEventRequest	7. The Test System responds with a TransactionEventResponse
<u>Note:</u> Steps 6 will report cost	

Tool validations
<p>* Step 2:</p> <p>Message SetDefaultTariffResponse</p> <ul style="list-style-type: none"> - status must be <i>Accepted</i> <p>* Step 4:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - eventType must be <i>Updated</i> - triggerReason must be <i>RunningCost</i> - transactionInfo.tariffId must be <i>Test System1</i> - costDetails.totalCost.currency must be <i>EUR</i> - costDetails.totalCost.chargingTime.exclTax must be <i>0.00</i> - costDetails.totalCost.chargingTime.inclTax must be <i>0.00</i> - costDetails.totalCost.fixed must be <i><omitted></i> - costDetails.totalCost.energy must be <i><omitted></i> - costDetails.totalCost.idleTime must be <i><omitted></i> - costDetails.totalCost.reservation must be <i><omitted></i> - costDetails.chargingPeriods must be <i><omitted></i> <p>* Step 6:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - eventType must be <i>Updated</i> - triggerReason must be <i>RunningCost</i> - transactionInfo.tariffId must be <i>Test System1</i> - costDetails.totalCost.currency must be <i>EUR</i> - costDetails.totalCost.chargingTime.exclTax must be greater <i>0.00</i> - costDetails.totalCost.chargingTime.inclTax must be greater <i>0.00</i> - costDetails.totalCost.fixed must be <i><omitted></i> - costDetails.totalCost.energy must be <i><omitted></i> - costDetails.totalCost.idleTime must be <i><omitted></i> - costDetails.totalCost.reservation must be <i><omitted></i> - costDetails.chargingPeriods must be <i><omitted></i>
<p>Post scenario validations:</p> <p>N/a</p>

TC_I_102_CS: Set Default Tariff - TariffMaxElements

Test case name	Set Default Tariff - TariffMaxElements
Test case Id	TC_I_102_CS
Use case Id(s)	I07
Requirement(s)	I07.FR.02, I07.FR.04
System under test	Charging Station
Description	To set the default tariff on the charging station, for example for ad hoc charging, or when there is no driver-specific tariff.
Purpose	To verify if the Charging Station correctly validates the tariff structure.
Prerequisite(s)	- The Charging Station supports local cost calculation using the Tariff mechanism (TariffCostCtrlr.Enabled[Tariff] exists and <i>ReadWrite</i> or is <i>true</i> and <i>ReadOnly</i>).

Before (Preparations)
Configuration State: - TariffCostCtrlr.Enabled[Tariff] is <i>true</i>
Memory State: N/a
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with GetVariablesResponse	1. The Test System sends a GetVariablesRequest with getVariableData[0].component.name = <i>TariffCostCtrlr</i> getVariableData[0].variable.name = <i>MaxElements</i> getVariableData[0].variable.instance = <i>Tariff</i>
<u>Note:</u> In the following steps, the value of <getVariableResult[0].attributeValue of step 2> is referred to as <i><TariffMaxElements></i> .	
4. The Charging Station responds with SetDefaultTariffResponse	3. The Test System sends a SetDefaultTariffRequest with evseld 0 tariff.tariffId <i>Test System1</i> tariff.currency <i>EUR</i> <i>repeat next <TariffMaxElements> +1 times with counter <i> starting at 0</i> tariff.energy.prices[i].priceKwh 1.00
6. The Charging Station responds with SetDefaultTariffResponse	5. The Test System sends a SetDefaultTariffRequest with evseld 0 tariff.tariffId <i>Test System1</i> tariff.currency <i>EUR</i> <i>repeat next <TariffMaxElements> times with counter <i> starting at 0</i> tariff.energy.prices[i].priceKwh 1.00
8. The Charging Station responds with SetDefaultTariffResponse	7. The Test System sends a SetDefaultTariffRequest with evseld 0 tariff.tariffId <i>Test System2</i> tariff.currency <i>EUR</i> <i>repeat next <TariffMaxElements> times with counter <i> starting at 0</i> tariff.energy.prices[i].priceKwh 1.00 <i>repeat next <TariffMaxElements> +1 times with counter <i> starting at 0</i> tariff.chargingTime.prices[i].priceMinute 1.00

Main (Test scenario)	
10. The Charging Station responds with SetDefaultTariffResponse	<p>9. The Test System sends a SetDefaultTariffRequest with evseld 0 tariff.tariffId Test System2 tariff.currency EUR <i>repeat next <TariffMaxElements> times with counter <i> starting at 0</i> tariff.energy.prices[i].priceKwh 1.00 <i>repeat next <TariffMaxElements> times with counter <i> starting at 0</i> tariff.chargingTime.prices[i].priceMinute 1.00</p>
12. The Charging Station responds with SetDefaultTariffResponse	<p>11. The Test System sends a SetDefaultTariffRequest with evseld 0 tariff.tariffId Test System3 tariff.currency EUR <i>repeat next <TariffMaxElements> times with counter <i> starting at 0</i> tariff.energy.prices[i].priceKwh 1.00 <i>repeat next <TariffMaxElements> times with counter <i> starting at 0</i> tariff.chargingTime.prices[i].priceMinute 1.00 <i>repeat next <TariffMaxElements> +1 times with counter <i> starting at 0</i> tariff.idleTime.prices[i].priceMinute 1.00</p>
14. The Charging Station responds with SetDefaultTariffResponse	<p>13. The Test System sends a SetDefaultTariffRequest with evseld 0 tariff.tariffId Test System3 tariff.currency EUR <i>repeat next <TariffMaxElements> times with counter <i> starting at 0</i> tariff.energy.prices[i].priceKwh 1.00 <i>repeat next <TariffMaxElements> times with counter <i> starting at 0</i> tariff.chargingTime.prices[i].priceMinute 1.00 <i>repeat next <TariffMaxElements> times with counter <i> starting at 0</i> tariff.idleTime.prices[i].priceMinute 1.00</p>
16. The Charging Station responds with SetDefaultTariffResponse	<p>15. The Test System sends a SetDefaultTariffRequest with evseld 0 tariff.tariffId Test System4 tariff.currency EUR <i>repeat next <TariffMaxElements> times with counter <i> starting at 0</i> tariff.energy.prices[i].priceKwh 1.00 <i>repeat next <TariffMaxElements> times with counter <i> starting at 0</i> tariff.chargingTime.prices[i].priceMinute 1.00 <i>repeat next <TariffMaxElements> times with counter <i> starting at 0</i> tariff.idleTime.prices[i].priceMinute 1.00 <i>repeat next <TariffMaxElements> +1 times with counter <i> starting at 0</i> tariff.fixedFee.prices[i].priceFixed 1.00</p>

Main (Test scenario)	
18. The Charging Station responds with SetDefaultTariffResponse	17. The Test System sends a SetDefaultTariffRequest with evseld 0 tariff.tariffId Test System4 tariff.currency EUR repeat next <TariffMaxElements> times with counter <i> starting at 0 tariff.energy.prices[i].priceKwh 1.00 repeat next <TariffMaxElements> times with counter <i> starting at 0 tariff.chargingTime.prices[i].priceMinute 1.00 repeat next <TariffMaxElements> times with counter <i> starting at 0 tariff.idleTime.prices[i].priceMinute 1.00 repeat next <TariffMaxElements> times with counter <i> starting at 0 tariff.fixedFee.prices[i].priceFixed 1.00
20. The Charging Station responds with SetDefaultTariffResponse	19. The Test System sends a SetDefaultTariffRequest with evseld 0 tariff.tariffId Test System4 tariff.currency EUR tariff.energy.prices[i].priceKwh 2.00

Tool validations
<p>* Step 2: Message GetVariablesResponse - getVariableResult[0].component.name = <i>TariffCostCtrlr</i> - getVariableResult[0].variable.name = <i>MaxElements</i> - getVariableResult[0].variable.instance <i>Tariff</i></p> <p>* Step 4: Message SetDefaultTariffResponse - status must be <i>TooManyElements</i></p> <p>* Step 6: Message SetDefaultTariffResponse - status must be <i>Accepted</i></p> <p>* Step 8: Message SetDefaultTariffResponse - status must be <i>TooManyElements</i></p> <p>* Step 10: Message SetDefaultTariffResponse - status must be <i>Accepted</i></p> <p>* Step 12: Message SetDefaultTariffResponse - status must be <i>TooManyElements</i></p> <p>* Step 14: Message SetDefaultTariffResponse - status must be <i>Accepted</i></p> <p>* Step 16: Message SetDefaultTariffResponse - status must be <i>TooManyElements</i></p> <p>* Step 18: Message SetDefaultTariffResponse - status must be <i>Accepted</i></p> <p>* Step 20: Message SetDefaultTariffResponse - status must be <i>DuplicateTariffId</i></p>

Tool validations
Post scenario validations: N/a

TC_I_103_CS: Set Default Tariff - CS doesn't support local cost calculation

Test case name	Set Default Tariff - CS doesn't support local cost calculation
Test case Id	TC_I_103_CS
Use case Id(s)	I07
Requirement(s)	I07.FR.01
System under test	Charging Station
Description	To set the default tariff on the charging station, for example for ad hoc charging, or when there is no driver-specific tariff.
Purpose	To verify if the Charging Station which doesn't support tariffs is responding like one which doesn't support it.
Prerequisite(s)	- The Charging Station doesn't support local cost calculation or has it disabled

Before (Preparations)

Configuration State:

- `TariffCostCtrlr.Available[Tariff]` is absent or *false*
- or `TariffCostCtrlr.Enabled[Tariff]` is *false*

Memory State:

N/a

Reusable State:

N/a

Main (Test scenario)

Charging Station	CSMS
2. The Charging Station responds with RPC Framework CALLERROR	1. The Test System sends a SetDefaultTariffRequest with evseld 0 tariff.tariffId <i>Test System1</i> tariff.currency <i>EUR</i> tariff.energy.taxRates[0].type <i>MyTax1</i> tariff.energy.taxRates[0].tax 20 tariff.energy.prices[0].priceKwh 1.00

Tool validations

* Step 2:

Message **CALLERROR**

- **ErrorCode** must be *NotSupported* or *NotImplemented*

Post scenario validations:

N/a

TC_I_104_CS: Set Default Tariff - transaction with default tariff

Test case name	Set Default Tariff - transaction with default tariff
Test case Id	TC_I_104_CS
Use case Id(s)	I07
Requirement(s)	I07.FR.10, I07.FR.14, I07.FR.32
System under test	Charging Station
Description	To set the default tariff on the charging station, for example for ad hoc charging, or when there is no driver-specific tariff.
Purpose	To verify if the Charging Station correctly reports the default tariff when a transaction is started.
Prerequisite(s)	- The Charging Station supports local cost calculation using the Tariff mechanism (TariffCostCtrlr.Enabled[Tariff] exists and <i>ReadWrite</i> or is <i>true</i> and <i>ReadOnly</i>).

Before (Preparations)
Configuration State: - TariffCostCtrlr.Enabled[Tariff] is <i>true</i>
Memory State: N/a
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with SetDefaultTariffResponse	1. The Test System sends a SetDefaultTariffRequest with evseld 0 tariff.tariffId Test System1 tariff.currency EUR tariff.energy.prices[0].priceKwh 2.00
3. Execute Reusable State <i>EnergyTransferStarted</i>	

Tool validations
* Step 2: Message SetDefaultTariffResponse - status must be <i>Accepted</i>
Post scenario validations: - One of the received TransactionEventRequest messages must contain transactionInfo.tariffId = <i>Test System1</i>

TC_I_105_CS: Set Default Tariff - TariffConditionsSupported is false

Test case name	Set Default Tariff - TariffConditionsSupported is false
Test case Id	TC_I_105_CS
Use case Id(s)	I07
Requirement(s)	I07.FR.03
System under test	Charging Station
Description	To set the default tariff on the charging station, for example for ad hoc charging, or when there is no driver-specific tariff.
Purpose	To verify if the Charging Station correctly validates the tariff structure.
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station supports local cost calculation using the Tariff mechanism (TariffCostCtrlr.Enabled[Tariff] exists and <i>ReadWrite</i> or is <i>true</i> and <i>ReadOnly</i>). - The Charging Station doesn't support tariff conditions (TariffCostCtrlr.ConditionsSupported[Tariff] is <i>false</i>)

Before (Preparations)
Configuration State: - TariffCostCtrlr.Enabled[Tariff] is <i>true</i>
Memory State: N/a
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS

Main (Test scenario)	
2. The Charging Station responds with SetDefaultTariffResponse	<p>1. The Test System sends a SetDefaultTariffRequest with evseld 0</p> <p>tariff.tariffId <i>Test System1</i></p> <p>tariff.currency <i>EUR</i></p> <p>tariff.energy.prices[0].priceKwh <i>1.00</i></p> <p>tariff.energy.prices[0].conditions.startTimeOfDay <i>00:00</i></p> <p>tariff.energy.prices[0].conditions.endTimeOfDay <i>00:00</i></p> <p>tariff.energy.prices[0].conditions.validFromDate <i>2012-01-01</i></p> <p>tariff.energy.prices[0].conditions.validToDate <i>2027-12-31</i></p> <p>tariff.energy.prices[0].conditions.minEnergy <i>10</i></p> <p>tariff.energy.prices[0].conditions.maxEnergy <i>10000</i></p> <p>tariff.energy.prices[0].conditions.minCurrent <i>1</i></p> <p>tariff.energy.prices[0].conditions.maxCurrent <i>10</i></p> <p>tariff.energy.prices[0].conditions.minPower <i>1000</i></p> <p>tariff.energy.prices[0].conditions.maxPower <i>10000</i></p> <p>tariff.energy.prices[0].conditions.minTime <i>0</i></p> <p>tariff.energy.prices[0].conditions.maxTime <i>3600</i></p> <p>tariff.energy.prices[0].conditions.minChargingTime <i>0</i></p> <p>tariff.energy.prices[0].conditions.maxChargingTime <i>3600</i></p> <p>tariff.energy.prices[0].conditions.minIdleTime <i>0</i></p> <p>tariff.energy.prices[0].conditions.maxIdleTime <i>3600</i></p> <p>tariff.energy.prices[0].conditions.dayOfWeek[0] <i>Monday</i></p> <p>tariff.energy.prices[0].conditions.dayOfWeek[1] <i>Tuesday</i></p> <p>tariff.energy.prices[0].conditions.dayOfWeek[2] <i>Wednesday</i></p> <p>tariff.energy.prices[0].conditions.dayOfWeek[3] <i>Thursday</i></p> <p>tariff.energy.prices[0].conditions.dayOfWeek[4] <i>Friday</i></p> <p>tariff.energy.prices[0].conditions.dayOfWeek[5] <i>Saturday</i></p> <p>tariff.energy.prices[0].conditions.dayOfWeek[6] <i>Sunday</i></p> <p>tariff.energy.prices[0].conditions.evseKind <i>AC</i></p> <p>tariff.fixedFee.prices[0].conditions.paymentBrand <i>PayMe</i></p> <p>tariff.fixedFee.prices[0].conditions.paymentRecognition <i>Debit</i></p>

Tool validations
<p>* Step 2:</p> <p>Message SetDefaultTariffResponse</p> <p>- status must be <i>ConditionNotSupported</i></p>
<p>Post scenario validations:</p> <p>N/a</p>

TC_I_106_CS: Set Default Tariff - validations

Test case name	Set Default Tariff - validations
Test case Id	TC_I_106_CS
Use case Id(s)	I07, I09
Requirement(s)	I07.FR.05, I07.FR.06, I07.FR.10, I07.FR.11, I07.FR.12, I07.FR.13, I09.FR.01, I09.FR.04
System under test	Charging Station
Description	To set the default tariff on the charging station, for example for ad hoc charging, or when there is no driver-specific tariff.
Purpose	To verify if the Charging Station correctly validates the tariff structure and persists after a reset.
Prerequisite(s)	- The Charging Station supports local cost calculation using the Tariff mechanism (TariffCostCtrlr.Enabled[Tariff] exists and <i>ReadWrite</i> or is <i>true</i> and <i>ReadOnly</i>).

Before (Preparations)
Configuration State: - TariffCostCtrlr.Enabled[Tariff] is <i>true</i>
Memory State: N/a
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with SetDefaultTariffResponse	1. The Test System sends a SetDefaultTariffRequest with evseld 0 tariff.tariffId <i>Test System1</i> tariff.currency <i>EUR</i> tariff.energy <omitted> tariff.chargingTime <omitted> tariff.idleTime <omitted> tariff.fixedFee <omitted>
4. The Charging Station responds with SetDefaultTariffResponse	3. The Test System sends a SetDefaultTariffRequest with evseld 0 tariff.tariffId <i>Test System1</i> tariff.currency <i>EUR</i> tariff.energy.prices[0].priceKwh <i>1.00</i> tariff.chargingTime <omitted> tariff.idleTime <omitted> tariff.fixedFee <omitted>
6. The Charging Station responds with SetDefaultTariffResponse	5. The Test System sends a SetDefaultTariffRequest with evseld 0 tariff.tariffId <i>Test System2</i> tariff.currency <i>EUR</i> tariff.energy <omitted> tariff.chargingTime.prices[0].priceMinute <i>1.00</i> tariff.idleTime <omitted> tariff.fixedFee <omitted>

Main (Test scenario)	
8. The Charging Station responds with SetDefaultTariffResponse	7. The Test System sends a SetDefaultTariffRequest with evseld 0 tariff.tariffId Test System3 tariff.currency EUR tariff.energy <omitted> tariff.chargingTime <omitted> tariff.idleTime.prices[0].priceMinute 1.00 tariff.fixedFee <omitted>
10. The Charging Station responds with SetDefaultTariffResponse	9. The Test System sends a SetDefaultTariffRequest with evseld 0 tariff.tariffId Test System4 tariff.currency EUR tariff.energy <omitted> tariff.chargingTime <omitted> tariff.idleTime <omitted> tariff.fixedFee.prices[0].priceFixed 1.00
12. The Charging Station responds with SetDefaultTariffResponse	11. The Test System sends a SetDefaultTariffRequest with evseld <Configured number of evse> + 1 tariff.tariffId Test System5 tariff.currency EUR tariff.energy.prices[0].priceKwh 1.00 tariff.chargingTime <omitted> tariff.idleTime <omitted> tariff.fixedFee <omitted>
14. The Charging Station responds with SetDefaultTariffResponse	13. The Test System sends a SetDefaultTariffRequest with evseld <Configured evseld> tariff.tariffId Test System5 tariff.currency EUR tariff.energy.prices[0].priceKwh 2.00 tariff.chargingTime <omitted> tariff.idleTime <omitted> tariff.fixedFee <omitted>
16. The Charging Station responds with GetTariffsResponse	15. The Test System sends a GetTariffsRequest with evseld 0
17. Execute Reusable State <i>Booted</i> with reset type <i>Immediate</i>	
19. The Charging Station responds with GetTariffsResponse	18. The Test System sends a GetTariffsRequest with evseld 0

Tool validations

* Step 2:

Message **SetDefaultTariffResponse**

- **status** must be *Rejected*
- **statusInfo.reasonCode** must be *<ommitted>* or *InvalidValue*

* Step 4:

Message **SetDefaultTariffResponse**

- **status** must be *Accepted*

* Step 6:

Message **SetDefaultTariffResponse**

- **status** must be *Accepted*

* Step 8:

Message **SetDefaultTariffResponse**

- **status** must be *Accepted*

* Step 10:

Message **SetDefaultTariffResponse**

- **status** must be *Accepted*

* Step 12:

Message **SetDefaultTariffResponse**

- **status** must be *Rejected*
- **statusInfo.reasonCode** must be *<ommitted>* or *UnknownEVSE*

* Step 14:

Message **SetDefaultTariffResponse**

- **status** must be *Accepted*

* Step 16:

Message **GetTariffsResponse**

- **status** must be *Accepted*

*Note: The following **tariffAssignments** do not need to be reported in specific order.*

IF *<Configured number of evse>* = 1

- **tariffAssignments[0].tariffId** must be *Test System5*
- **tariffAssignments[0].tariffKind** must be *DefaultTariff*
- **tariffAssignments[0].evselds[0]** must be *<Configured evseld>*

END IF

IF *<Configured number of evse>* > 1

- **tariffAssignments[0].tariffId** must be *Test System5*
- **tariffAssignments[0].tariffKind** must be *DefaultTariff*
- **tariffAssignments[0].evselds[0]** must be *<Configured evseld>*
- **tariffAssignments[1].tariffId** must be *Test System4*
- **tariffAssignments[1].tariffKind** must be *DefaultTariff*
- **tariffAssignments[1].evselds[*]** must contain all evselds up to *<Configured number of evse>* with exception of the *<Configured evseld>*

END IF

Tool validations
<p>* Step 19:</p> <p>Message GetTariffsResponse</p> <ul style="list-style-type: none">- status must be <i>Accepted</i> <p>IF <Configured number of evse> = 1</p> <ul style="list-style-type: none">- tariffAssignments[0].tariffId must be <i>Test System5</i>- tariffAssignments[0].tariffKind must be <i>DefaultTariff</i>- tariffAssignments[0].evselds[0] must be <i><Configured evseld></i> <p>END IF</p> <p>IF <Configured number of evse> > 1</p> <ul style="list-style-type: none">- tariffAssignments[0].tariffId must be <i>Test System5</i>- tariffAssignments[0].tariffKind must be <i>DefaultTariff</i>- tariffAssignments[0].evselds[0] must be <i><Configured evseld></i>- tariffAssignments[1].tariffId must be <i>Test System4</i>- tariffAssignments[1].tariffKind must be <i>DefaultTariff</i>- tariffAssignments[1].evselds[*] must contain all evselds up to <i><Configured number of evse></i> with exception of the <i><Configured evseld></i> <p>END IF</p>
<p>Post scenario validations:</p> <p>N/a</p>

TC_I_107_CS: Receive Driver Tariff - CS cannot process tariff - UseDefault/CentralCost

Test case name	Receive Driver Tariff - CS cannot process tariff - UseDefault/CentralCost
Test case Id	TC_I_107_CS
Use case Id(s)	I08
Requirement(s)	I08.FR.30, I08.FR.32, I08.FR.33
System under test	Charging Station
Description	To support receiving the driver-specific tariff to enable local cost calculation based on a tariff for this driver.
Purpose	To verify if the Charging Station which doesn't support local cost calculation properly handles receiving a tariff in AuthorizeResponse.
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station doesn't support tariffs or has it disabled - The Charging Station is able to send AuthorizeRequest for local authorization (RFID) or remote start

Before (Preparations)
Configuration State: <ul style="list-style-type: none"> - AuthCtrlr.Enabled = true - AuthCtrlr.AuthorizeRemoteStart = true - TariffCostCtrlr.Available[Tariff] is false - or TariffCostCtrlr.Enabled[Tariff] is false - TariffCostCtrlr.HandleFailedTariff is UseDefault or CentralCost
Memory State: N/a
Reusable State: State EVConnectedPreSession

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with SetDefaultTariffResponse	1. The Test System sends a SetDefaultTariffRequest with evseld 0 tariff.tariffId Test System1A tariff.currency EUR tariff.energy.prices[0].priceKwh 1.00
Option A: If only local authorization is supported: <u>Manual Action:</u> <i>Present idToken.</i>	
Option B: If remote starting is supported: 4. The Charging Station sends an RequestStartTransactionResponse	3. The Test System sends an RequestStartTransactionRequest with idToken = <configured valid_idtoken_idtoken>
5. The Charging Station sends an AuthorizeRequest	6. The Test System responds with an AuthorizeResponse with idTokenInfo.status Accepted tariff.tariffId Test System1 tariff.currency EUR tariff.energy.prices[0].priceKwh 2.00
<u>Note:</u> A transaction may have been started already when TxStartPoint contains <i>ParkingBayOccupancy</i> or <i>EVConnected</i> . Energy shall be delivered, because <i>idToken</i> is Accepted and Charging Station continues with default tariff or CSMS does central cost calculation	
7. The Charging Station sends a NotifyEventRequest	8. The Test System sends a NotifyEventResponse

Tool validations
<p>* Step 2:</p> <p>Message SetDefaultTariffResponse</p> <ul style="list-style-type: none">- status <i>Accepted</i> <p>* Step 7:</p> <p>Message NotifyEventRequest</p> <ul style="list-style-type: none">- trigger = <i>Alerting</i> or <i>Delta</i>- component = <i>TariffCostCtrlr</i>- variable = <i>Problem</i>- actualValue = <i>true</i>
<p>Post scenario validations:</p> <ul style="list-style-type: none">- Charging Station SHALL deliver energy.

TC_I_108_CS: Receive Driver Tariff - CS cannot process tariff - Deauthorize

Test case name	Receive Driver Tariff - CS cannot process tariff - Deauthorize
Test case Id	TC_I_108_CS
Use case Id(s)	I08
Requirement(s)	I08.FR.30, I08.FR.31
System under test	Charging Station
Description	To support receiving the driver-specific tariff to enable local cost calculation based on a tariff for this driver.
Purpose	To verify if the Charging Station which doesn't support local cost calculation properly handles receiving a tariff in AuthorizeResponse.
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station doesn't support tariffs or has it disabled - The Charging Station is able to send AuthorizeRequest for local authorization (RFID) or remote start

Before (Preparations)
Configuration State: <ul style="list-style-type: none"> - AuthCtrlr.Enabled = <i>true</i> - AuthCtrlr.AuthorizeRemoteStart = <i>true</i> - TariffCostCtrlr.Available[Tariff] is absent or <i>false</i> - or TariffCostCtrlr.Enabled[Tariff] is <i>false</i> - TariffCostCtrlr.HandleFailedTariff is <i>Deauthorize</i>
Memory State: N/a
Reusable State: State <i>EVConnectedPreSession</i>

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>Authorized</i> AuthorizeResponse with: idTokenInfo.status <i>Accepted</i> tariff.tariffId <i>Test System1</i> tariff.currency <i>EUR</i> tariff.energy.prices[0].priceKwh <i>2.00</i>	
2. The Charging Station sends a NotifyEventRequest	3. The Test System sends a NotifyEventResponse
4. The Charging Stations does NOT send a TransactionEventRequest with ChargingState is <i>Charging</i>	5. The Test System responds with a TransactionEventResponse

Tool validations
* Step 2: Message NotifyEventRequest <ul style="list-style-type: none"> - trigger = <i>Alerting</i> or <i>Delta</i> - component = <i>TariffCostCtrlr</i> - variable = <i>Problem</i> - actualValue = <i>true</i> <i>Note: It is also allowed to receive other NotifyEventRequests from the Charging Station.</i>
Post scenario validations: <ul style="list-style-type: none"> - Charging Station SHALL not deliver energy.

TC_I_109_CS: Receive Driver Tariff - Goodflow

Test case name	Receive Driver Tariff - Goodflow
Test case Id	TC_I_109_CS
Use case Id(s)	I08, I09
Requirement(s)	I08.FR.04, I08.FR.05, I08.FR.06, I08.FR.07, I09.FR.03
System under test	Charging Station
Description	To support receiving the driver-specific tariff to enable local cost calculation based on a tariff for this driver.
Purpose	To verify if the Charging Station supports driver tariffs.
Prerequisite(s)	- The Charging Station supports local cost calculation using the Tariff mechanism (TariffCostCtrlr.Enabled[Tariff] exists and <i>ReadWrite</i> or is <i>true</i> and <i>ReadOnly</i>).

Before (Preparations)
Configuration State: - TxCtrlr.EVConnectionTimeout is <Configured ev_connection_timeout> - AuthCtrlr.Enabled = <i>true</i> - AuthCtrlr.AuthorizeRemoteStart = <i>true</i> - TariffCostCtrlr.Enabled[Tariff] is <i>true</i>
Memory State: N/a
Reusable State: State is <i>EVConnectedPreSession</i>

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>Authorized</i> AuthorizeResponse with: idTokenInfo.status Accepted tariff.tariffId Test System1 tariff.currency EUR tariff.energy.prices[0].priceKwh 2.00	
Manual Action: Accept tariff on Charging Station (This could be automatically accepted).	
2. The Charging Station sends a TransactionEventRequest	3. The Test System responds with a TransactionEventResponse
5. The Charging Station responds with GetTariffsResponse	4. The Test System sends a GetTariffsRequest with evseld <Configured evseld>
6. Execute Reusable State <i>StopAuthorized</i> Execute Reusable State <i>EVDIsconnected</i> Execute Reusable State <i>ParkingBayUnoccupied</i>	
8. The Charging Station responds with GetTariffsResponse	7. The Test System sends a GetTariffsRequest with evseld <Configured evseld>
9. Execute Reusable State <i>Authorized</i> AuthorizeResponse with: idTokenInfo.status Accepted tariff.tariffId Test System1 tariff.currency EUR tariff.energy.prices[0].priceKwh 2.00	
The Test System waits <Configured EV connection timeout> seconds.	

Main (Test scenario)	
10. The Charging Station sends a <code>TransactionEventRequest</code> <u>Note(s):</u> - This step needs to be executed after the <code><Configured ev_connection_timeout></code> expires, if the transaction has been started. So in the case <code>TxStartPoint</code> contains <code>ParkingBayOccupancy</code> OR <code>Authorized</code>	11. The Test System responds with a <code>TransactionEventResponse</code>
13. The Charging Station responds with <code>GetTariffsResponse</code>	12. The Test System sends a <code>GetTariffsRequest</code> with <code>evseld</code> <code><Configured evseld></code>

Tool validations
<p>* Step 2: Message <code>TransactionEventRequest</code> - <code>transactionInfo.tariffId</code> must be <i>Test System1</i></p> <p>* Step 5: Message <code>GetTariffsResponse</code> - <code>status</code> must be <i>Accepted</i> - <code>tariffAssignments[0].tariffId</code> must be <i>Test System1</i> - <code>tariffAssignments[0].tariffKind</code> must be <i>DriverTariff</i> - <code>tariffAssignments[0].idTokens[0]</code> must be <i><Configured valid_idtoken_idtoken></i></p> <p>* Step 8: Message <code>GetTariffsResponse</code> - <code>status</code> must be <i>NoTariff</i></p> <p>* Step 10: Message: <code>TransactionEventRequest</code> - <code>triggerReason</code> must be <i>EVConnectTimeout</i> - <code>transactionInfo.stoppedReason</code> must be <i>Timeout</i></p> <p>* Step 13: Message <code>GetTariffsResponse</code> - <code>status</code> must be <i>NoTariff</i></p> <p>Post scenario validations: N/a</p>

TC_I_110_CS: Clear Tariffs - DefaultTariff

Test case name	Clear Tariffs - DefaultTariff
Test case Id	TC_I_110_CS
Use case Id(s)	I10
Requirement(s)	I10.FR.02, I10.FR.03, I10.FR.06
System under test	Charging Station
Description	To clear previously set tariffs on the charging station.
Purpose	To verify if the Charging Station correctly clearing tariffs.
Prerequisite(s)	- The Charging Station supports local cost calculation using the Tariff mechanism (TariffCostCtrlr.Enabled[Tariff] exists and <i>ReadWrite</i> or is <i>true</i> and <i>ReadOnly</i>).

Before (Preparations)
Configuration State: - TariffCostCtrlr.Enabled[Tariff] is <i>true</i>
Memory State: N/a
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with SetDefaultTariffResponse	1. The Test System sends a SetDefaultTariffRequest with evseld 0 tariff.tariffId Test System1A tariff.currency EUR energy.prices[0].priceKwh 0.10
4. The Charging Station responds with SetDefaultTariffResponse	3. The Test System sends a SetDefaultTariffRequest with evseld 0 tariff.tariffId Test System1B tariff.validFrom <Current DateTime + 1:00 hour> tariff.currency EUR energy.prices[0].priceKwh 0.20
6. The Charging Station responds with SetDefaultTariffResponse	5. The Test System sends a SetDefaultTariffRequest with evseld 0 tariff.tariffId Test System1C tariff.validFrom <Current DateTime + 2:00 hour> tariff.currency EUR energy.prices[0].priceKwh 0.30
8. The Charging Station responds with SetDefaultTariffResponse	7. The Test System sends a SetDefaultTariffRequest with evseld 0 tariff.tariffId Test System1D tariff.validFrom <Current DateTime + 3:00 hour> tariff.currency EUR energy.prices[0].priceKwh 0.40
10. The Charging Station responds with ClearTariffsResponse	9. The Test System sends a ClearTariffsRequest with tariffIds[0] Test System1B tariffIds[1] Test System1A
12. The Charging Station responds with ClearTariffsResponse	11. The Test System sends a ClearTariffsRequest with tariffIds <omitted>

Tool validations
<p>* Step 2:</p> <p>Message SetDefaultTariffResponse</p> <ul style="list-style-type: none"> - status must be <i>Accepted</i> <p>* Step 4:</p> <p>Message SetDefaultTariffResponse</p> <ul style="list-style-type: none"> - status must be <i>Accepted</i> <p>* Step 6:</p> <p>Message SetDefaultTariffResponse</p> <ul style="list-style-type: none"> - status must be <i>Accepted</i> <p>* Step 8:</p> <p>Message SetDefaultTariffResponse</p> <ul style="list-style-type: none"> - status must be <i>Accepted</i> <p>* Step 10:</p> <p>Message ClearTariffsResponse</p> <p><i>Note: The following clearTariffsResults do not need to be reported in specific order.</i></p> <ul style="list-style-type: none"> - clearTariffsResult[0].tariffId must be <i>Test System1B</i> - clearTariffsResult[0].status must be <i>Accepted</i> - clearTariffsResult[1].tariffId must be <i>Test System1A</i> - clearTariffsResult[1].status must be <i>Accepted</i> <p>* Step 12:</p> <p>Message ClearTariffsResponse</p> <p><i>Note: The following clearTariffsResults do not need to be reported in specific order.</i></p> <ul style="list-style-type: none"> - clearTariffsResult[0].tariffId must be <i>Test System1C</i> - clearTariffsResult[0].status must be <i>Accepted</i> - clearTariffsResult[1].tariffId must be <i>Test System1D</i> - clearTariffsResult[1].status must be <i>Accepted</i>
<p>Post scenario validations:</p> <p>N/a</p>

TC_I_111_CS: Clear Tariffs - Tariff in use

Test case name	Clear Tariffs - Tariff in use
Test case Id	TC_I_111_CS
Use case Id(s)	I10, I07
Requirement(s)	I10.FR.01, I10.FR.07, I07.FR.10
System under test	Charging Station
Description	To clear previously set tariffs on the charging station.
Purpose	To verify if the Charging Station correctly clearing tariffs.
Prerequisite(s)	- The Charging Station supports local cost calculation using the Tariff mechanism (TariffCostCtrlr.Enabled[Tariff] exists and <i>ReadWrite</i> or is <i>true</i> and <i>ReadOnly</i>).

Before (Preparations)
Configuration State: - TariffCostCtrlr.Enabled[Tariff] is <i>true</i>
Memory State: N/a
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with SetDefaultTariffResponse	1. The Test System sends a SetDefaultTariffRequest with evseld 0 tariff.tariffId <i>Test System1</i> tariff.currency <i>EUR</i> tariff.energy.prices[0].priceKwh <i>2.00</i>
3. Execute Reusable State <i>EnergyTransferStarted</i>	
5. The Charging Station responds with ClearTariffsResponse	4. The Test System sends a ClearTariffsRequest with tariffIds <i><omitted></i> tariffKind <i><omitted></i>
7. The Charging Station responds with GetTariffsResponse	6. The Test System sends a GetTariffsRequest with evseld 0

Tool validations
* Step 2: Message SetDefaultTariffResponse - status must be <i>Accepted</i> * Step 5: Message ClearTariffsResponse - clearTariffsResult[0].tariffId must be <i>Test System1</i> - clearTariffsResult[0].status must be <i>Accepted</i> * Step 7: Message GetTariffsResponse - status must be <i>Accepted</i> - tariffAssignments[0].tariffId must be <i>Test System1</i> - tariffAssignments[0].tariffKind must be <i>DefaultTariff</i>
Post scenario validations: N/a

TC_I_112_CS: Local Cost Calculation - Change transaction tariff - local cost calculation unsupported

Test case name	Local Cost Calculation - Change transaction tariff - local cost calculation unsupported
Test case Id	TC_I_112_CS
Use case Id(s)	I11
Requirement(s)	I11.FR.01
System under test	Charging Station
Description	CSMS changes the tariff that is associated with a transaction. This may be needed when dealing with unexpected price changes.
Purpose	To verify if the Charging Station correctly validates the tariff change.
Prerequisite(s)	- The Charging Station doesn't support local cost calculation or has it disabled.

Before (Preparations)
Configuration State: - <code>TariffCostCtrlr.Available[Tariff]</code> is absent or <i>false</i> - or <code>TariffCostCtrlr.Enabled[Tariff]</code> is absent or <i>false</i>
Memory State: N/a
Reusable State: State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with RPC Framework CALLERROR	1. The Test System sends a ChangeTransactionTariffRequest with transactionId 0 tariff.tariffId Test System1 tariff.currency EUR tariff.energy.prices[0].priceKwh 1.00

Tool validations
* Step 2: Message CALLERROR - ErrorCode must be <i>NotSupported</i> or <i>NotImplemented</i>
Post scenario validations: N/a

TC_I_113_CS: Local Cost Calculation - Change transaction tariff - TariffMaxElements

Test case name	Local Cost Calculation - Change transaction tariff - TariffMaxElements
Test case Id	TC_I_113_CS
Use case Id(s)	I11
Requirement(s)	I11.FR.02
System under test	Charging Station
Description	CSMS changes the tariff that is associated with a transaction. This may be needed when dealing with unexpected price changes.
Purpose	To verify if the Charging Station correctly validates the tariff change.
Prerequisite(s)	- The Charging Station supports local cost calculation using the Tariff mechanism (TariffCostCtrlr.Enabled[Tariff] exists and <i>ReadWrite</i> or is <i>true</i> and <i>ReadOnly</i>). This testcase is only executable if MaxElements is <= 1000

Before (Preparations)
Configuration State: - TariffCostCtrlr.Enabled[Tariff] is <i>true</i>
Memory State: N/a
Reusable State: State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with GetVariablesResponse	1. The Test System sends a GetVariablesRequest with getVariableData[0].component.name = <i>TariffCostCtrlr</i> getVariableData[0].variable.name = <i>MaxElements</i> getVariableData[0].variable.instance = <i>Tariff</i>
<u>Note:</u> In the following steps, the value of <getVariableResult[0].attributeValue of step 2> is referred to as <TariffMaxElements> .	
4. The Charging Station responds with ChangeTransactionTariffResponse	3. The Test System sends a ChangeTransactionTariffRequest with transactionId <transaction id> tariff.tariffId <i>Test System1</i> tariff.currency <i>EUR</i> repeat next <TariffMaxElements> +1 times with counter <i> starting at 0 tariff.energy.prices[i].priceKwh <i>1.00</i>
6. The Charging Station responds with ChangeTransactionTariffResponse	5. The Test System sends a ChangeTransactionTariffRequest with transactionId <transaction id> tariff.tariffId <i>Test System1</i> tariff.currency <i>EUR</i> repeat next <TariffMaxElements> times with counter <i> starting at 0 tariff.energy.prices[i].priceKwh <i>1.00</i>

Main (Test scenario)	
8. The Charging Station responds with ChangeTransactionTariffResponse	<p>7. The Test System sends a ChangeTransactionTariffRequest with</p> <p>transactionId <transaction id></p> <p>tariff.tariffId Test System2</p> <p>tariff.currency EUR</p> <p>repeat next <TariffMaxElements> times with counter <i> starting at 0</p> <p>tariff.energy.prices[i].priceKwh 1.00</p> <p>repeat next <TariffMaxElements> +1 times with counter <i> starting at 0</p> <p>tariff.chargingTime.prices[i].priceMinute 1.00</p>
10. The Charging Station responds with ChangeTransactionTariffResponse	<p>9. The Test System sends a ChangeTransactionTariffRequest with</p> <p>transactionId <transaction id></p> <p>tariff.tariffId Test System2</p> <p>tariff.currency EUR</p> <p>repeat next <TariffMaxElements> times with counter <i> starting at 0</p> <p>tariff.energy.prices[i].priceKwh 1.00</p> <p>repeat next <TariffMaxElements> times with counter <i> starting at 0</p> <p>tariff.chargingTime.prices[i].priceMinute 1.00</p>
12. The Charging Station responds with ChangeTransactionTariffResponse	<p>11. The Test System sends a ChangeTransactionTariffRequest with</p> <p>transactionId <transaction id></p> <p>tariff.tariffId Test System3</p> <p>tariff.currency EUR</p> <p>repeat next <TariffMaxElements> times with counter <i> starting at 0</p> <p>tariff.energy.prices[i].priceKwh 1.00</p> <p>repeat next <TariffMaxElements> times with counter <i> starting at 0</p> <p>tariff.chargingTime.prices[i].priceMinute 1.00</p> <p>repeat next <TariffMaxElements> +1 times with counter <i> starting at 0</p> <p>tariff.idleTime.prices[i].priceMinute 1.00</p>
14. The Charging Station responds with ChangeTransactionTariffResponse	<p>13. The Test System sends a ChangeTransactionTariffRequest with</p> <p>transactionId <transaction id></p> <p>tariff.tariffId Test System3</p> <p>tariff.currency EUR</p> <p>repeat next <TariffMaxElements> times with counter <i> starting at 0</p> <p>tariff.energy.prices[i].priceKwh 1.00</p> <p>repeat next <TariffMaxElements> times with counter <i> starting at 0</p> <p>tariff.chargingTime.prices[i].priceMinute 1.00</p> <p>repeat next <TariffMaxElements> times with counter <i> starting at 0</p> <p>tariff.idleTime.prices[i].priceMinute 1.00</p>

Main (Test scenario)	
16. The Charging Station responds with ChangeTransactionTariffResponse	<p>15. The Test System sends a ChangeTransactionTariffRequest with</p> <p>transactionId <transaction id></p> <p>tariff.tariffId Test System4</p> <p>tariff.currency EUR</p> <p>repeat next <TariffMaxElements> times with counter <i> starting at 0</p> <p>tariff.energy.prices[i].priceKwh 1.00</p> <p>repeat next <TariffMaxElements> times with counter <i> starting at 0</p> <p>tariff.chargingTime.prices[i].priceMinute 1.00</p> <p>repeat next <TariffMaxElements> times with counter <i> starting at 0</p> <p>tariff.idleTime.prices[i].priceMinute 1.00</p> <p>repeat next <TariffMaxElements> +1 times with counter <i> starting at 0</p> <p>tariff.fixedFee.prices[i].priceFixed 1.00</p>
18. The Charging Station responds with ChangeTransactionTariffResponse	<p>17. The Test System sends a ChangeTransactionTariffRequest with</p> <p>transactionId <transaction id></p> <p>tariff.tariffId Test System4</p> <p>tariff.currency EUR</p> <p>repeat next <TariffMaxElements> times with counter <i> starting at 0</p> <p>tariff.energy.prices[i].priceKwh 1.00</p> <p>repeat next <TariffMaxElements> times with counter <i> starting at 0</p> <p>tariff.chargingTime.prices[i].priceMinute 1.00</p> <p>repeat next <TariffMaxElements> times with counter <i> starting at 0</p> <p>tariff.idleTime.prices[i].priceMinute 1.00</p> <p>repeat next <TariffMaxElements> times with counter <i> starting at 0</p> <p>tariff.fixedFee.prices[i].priceFixed 1.00</p>

Tool validations
<div><div>* Step 2:</div><div>Message GetVariablesResponse</div><div><div>- getVariableResult[0].component.name = <i>TariffCostCtrlr</i></div><div>- getVariableResult[0].variable.name = <i>MaxElements</i></div><div>- getVariableResult[0].variable.instance <i>Tariff</i></div></div><div><div>* Step 4:</div><div>Message ChangeTransactionTariffResponse</div><div><div>- status must be <i>TooManyElements</i></div></div><div><div>* Step 6:</div><div>Message ChangeTransactionTariffResponse</div><div><div>- status must be <i>Accepted</i></div></div><div><div>* Step 8:</div><div>Message ChangeTransactionTariffResponse</div><div><div>- status must be <i>TooManyElements</i></div></div><div><div>* Step 10:</div><div>Message ChangeTransactionTariffResponse</div><div><div>- status must be <i>Accepted</i></div></div><div><div>* Step 12:</div><div>Message ChangeTransactionTariffResponse</div><div><div>- status must be <i>TooManyElements</i></div></div><div><div>* Step 14:</div><div>Message ChangeTransactionTariffResponse</div><div><div>- status must be <i>Accepted</i></div></div><div><div>* Step 16:</div><div>Message ChangeTransactionTariffResponse</div><div><div>- status must be <i>TooManyElements</i></div></div><div><div>* Step 18:</div><div>Message ChangeTransactionTariffResponse</div><div><div>- status must be <i>Accepted</i></div></div></div></div></div></div></div></div></div></div></div>
<div><div>Post scenario validations:</div><div>N/a</div></div>

TC_I_114_CS: Local Cost Calculation - Change transaction tariff - TariffConditionsSupported is false

Test case name	Local Cost Calculation - Change transaction tariff - TariffConditionsSupported is false
Test case Id	TC_I_114_CS
Use case Id(s)	I11
Requirement(s)	I11.FR.03
System under test	Charging Station
Description	CSMS changes the tariff that is associated with a transaction. This may be needed when dealing with unexpected price changes.
Purpose	To verify if the Charging Station correctly validates the tariff change.
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station supports local cost calculation using the Tariff mechanism (TariffCostCtrlr.Enabled[Tariff] exists and <i>ReadWrite</i> or is <i>true</i> and <i>ReadOnly</i>). - The Charging Station doesn't support tariff conditions (TariffCostCtrlr.ConditionsSupported[Tariff] is <i>false</i>)

Before (Preparations)
Configuration State: <ul style="list-style-type: none"> - TariffCostCtrlr.Enabled[Tariff] is <i>true</i> - TariffCostCtrlr.ConditionsSupported[Tariff] is <i>false</i>
Memory State: N/a
Reusable State: State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS

Main (Test scenario)	
2. The Charging Station responds with ChangeTransactionTariffResponse	<p>1. The Test System sends a ChangeTransactionTariffRequest with</p> <pre> transactionId <transaction id> tariff.tariffId Test System99 tariff.currency EUR tariff.energy.prices[0].priceKwh 1.00 tariff.energy.prices[0].conditions.startTimeOfDay 00:00 tariff.energy.prices[0].conditions.endTimeOfDay 00:00 tariff.energy.prices[0].conditions.validFromDate 2012-01-01 tariff.energy.prices[0].conditions.validToDate 2027-12-31 tariff.energy.prices[0].conditions.minEnergy 10 tariff.energy.prices[0].conditions.maxEnergy 10000 tariff.energy.prices[0].conditions.minCurrent 1 tariff.energy.prices[0].conditions.maxCurrent 10 tariff.energy.prices[0].conditions.minPower 1000 tariff.energy.prices[0].conditions.maxPower 10000 tariff.energy.prices[0].conditions.minTime 0 tariff.energy.prices[0].conditions.maxTime 3600 tariff.energy.prices[0].conditions.minChargingTime 0 tariff.energy.prices[0].conditions.maxChargingTime 3600 tariff.energy.prices[0].conditions.minIdleTime 0 tariff.energy.prices[0].conditions.maxIdleTime 3600 tariff.energy.prices[0].conditions.dayOfWeek[0] Monday tariff.energy.prices[0].conditions.dayOfWeek[1] Tuesday tariff.energy.prices[0].conditions.dayOfWeek[2] Wednesday tariff.energy.prices[0].conditions.dayOfWeek[3] Thursday tariff.energy.prices[0].conditions.dayOfWeek[4] Friday tariff.energy.prices[0].conditions.dayOfWeek[5] Saturday tariff.energy.prices[0].conditions.dayOfWeek[6] Sunday tariff.energy.prices[0].conditions.evseKind AC tariff.fixedFee.prices[0].conditions.paymentBrand PayMe tariff.fixedFee.prices[0].conditions.paymentRecognition Debit </pre>

Tool validations
<p>* Step 2:</p> <p>Message ChangeTransactionTariffResponse</p> <p>- status must be <i>ConditionNotSupported</i></p>
<p>Post scenario validations:</p> <p>N/a</p>

TC_I_115_CS: Local Cost Calculation - Change transaction tariff - TariffConditionsSupported is true

Test case name	Local Cost Calculation - Change transaction tariff - TariffConditionsSupported is true
Test case Id	TC_I_115_CS
Use case Id(s)	I11
Requirement(s)	I11.FR.03, I11.FR.06, I11.FR.07
System under test	Charging Station
Description	CSMS changes the tariff that is associated with a transaction. This may be needed when dealing with unexpected price changes.
Purpose	To verify if the Charging Station correctly validates the tariff change.
Prerequisite(s)	- The Charging Station supports local cost calculation using the Tariff mechanism (TariffCostCtrlr.Enabled[Tariff] exists and ReadWrite or is true and ReadOnly). - The Charging Station supports tariff conditions (TariffCostCtrlr.ConditionsSupported[Tariff] is true)

Before (Preparations)
Configuration State: - TariffCostCtrlr.Enabled[Tariff] is true - TariffCostCtrlr.ConditionsSupported[Tariff] is true
Memory State: N/a
Reusable State: State is EnergyTransferStarted

Main (Test scenario)
Charging StationCSMS

Main (Test scenario)	
2. The Charging Station responds with ChangeTransactionTariffResponse	<p>1. The Test System sends a ChangeTransactionTariffRequest with</p> <p>transactionId <transaction id></p> <p>tariff.tariffId <i>Test System99</i></p> <p>tariff.currency <i>EUR</i></p> <p>tariff.energy.prices[0].priceKwh <i>1.00</i></p> <p>tariff.energy.prices[0].conditions.startTimeOfDay <i>00:00</i></p> <p>tariff.energy.prices[0].conditions.endTimeOfDay <i>00:00</i></p> <p>tariff.energy.prices[0].conditions.validFromDate <i>2012-01-01</i></p> <p>tariff.energy.prices[0].conditions.validToDate <i>2027-12-31</i></p> <p>tariff.energy.prices[0].conditions.minEnergy <i>10</i></p> <p>tariff.energy.prices[0].conditions.maxEnergy <i>10000</i></p> <p>tariff.energy.prices[0].conditions.minCurrent <i>1</i></p> <p>tariff.energy.prices[0].conditions.maxCurrent <i>10</i></p> <p>tariff.energy.prices[0].conditions.minPower <i>1000</i></p> <p>tariff.energy.prices[0].conditions.maxPower <i>10000</i></p> <p>tariff.energy.prices[0].conditions.minTime <i>0</i></p> <p>tariff.energy.prices[0].conditions.maxTime <i>3600</i></p> <p>tariff.energy.prices[0].conditions.minChargingTime <i>0</i></p> <p>tariff.energy.prices[0].conditions.maxChargingTime <i>3600</i></p> <p>tariff.energy.prices[0].conditions.minIdleTime <i>0</i></p> <p>tariff.energy.prices[0].conditions.maxIdleTime <i>3600</i></p> <p>tariff.energy.prices[0].conditions.dayOfWeek[0] <i>Monday</i></p> <p>tariff.energy.prices[0].conditions.dayOfWeek[1] <i>Tuesday</i></p> <p>tariff.energy.prices[0].conditions.dayOfWeek[2] <i>Wednesday</i></p> <p>tariff.energy.prices[0].conditions.dayOfWeek[3] <i>Thursday</i></p> <p>tariff.energy.prices[0].conditions.dayOfWeek[4] <i>Friday</i></p> <p>tariff.energy.prices[0].conditions.dayOfWeek[5] <i>Saturday</i></p> <p>tariff.energy.prices[0].conditions.dayOfWeek[6] <i>Sunday</i></p> <p>tariff.energy.prices[0].conditions.evseKind <i>AC</i></p> <p>tariff.fixedFee.prices[0].conditions.paymentBrand <i>PayMe</i></p> <p>tariff.fixedFee.prices[0].conditions.paymentRecognition <i>Debit</i></p>
3. The Charging Station sends a TransactionEventRequest	4. The Test System responds with a TransactionEventResponse

Tool validations
<p>* Step 2:</p> <p>Message ChangeTransactionTariffResponse</p> <ul style="list-style-type: none"> - status must be <i>Accepted</i> <p>* Step 3:</p> <p>Message TransactionEventRequest</p> <ul style="list-style-type: none"> - triggerReason must be <i>TariffChanged</i> - transactionInfo.tariffId must be <i>Test System99</i>
<p>Post scenario validations:</p> <p>N/a</p>

TC_I_116_CS: Local Cost Calculation - Change transaction tariff - goodflow

Test case name	Local Cost Calculation - Change transaction tariff - goodflow
Test case Id	TC_I_116_CS
Use case Id(s)	I11
Requirement(s)	I11.FR.06, I11.FR.08
System under test	Charging Station
Description	CSMS changes the tariff that is associated with a transaction. This may be needed when dealing with unexpected price changes.
Purpose	To verify if the Charging Station correctly validates the tariff change.
Prerequisite(s)	- The Charging Station supports local cost calculation using the Tariff mechanism (TariffCostCtrlr.Enabled[Tariff] exists and <i>ReadWrite</i> or is <i>true</i> and <i>ReadOnly</i>).

Before (Preparations)
Configuration State: - TariffCostCtrlr.Enabled[Tariff] is <i>true</i> - TariffCostCtrlr.Enabled[RunningCost] is <i>true</i> - TariffCostCtrlr.Enabled[Cost] is <i>true</i> - TariffCostCtrlr.Interval[Cost] is 15 - SampledDataCtrl.TxUpdatedInterval = 0
Memory State: N/a
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with SetDefaultTariffResponse	1. The Test System sends a SetDefaultTariffRequest with evseld 0 tariff.tariffId <i>Test System1</i> tariff.currency <i>EUR</i> tariff.chargingTime.prices[0].priceMinute <i>0.10</i> tariff.chargingTime.taxRates[0].tax 0 tariff.chargingTime.taxRates[0].type <i>VAT</i>
3. Execute Reusable State <i>EnergyTransferStarted</i>	
4. The Charging Station sends a TransactionEventRequest	5. The Test System responds with a TransactionEventResponse
<i>Note: Wait 1 minute. Steps 4 and 5 will be repeated every 15 seconds</i>	
6. The Charging Station sends a TransactionEventRequest	7. The Test System responds with a TransactionEventResponse
9. The Charging Station responds with ChangeTransactionTariffResponse	8. The Test System sends a ChangeTransactionTariffRequest with transactionId <i><transaction id></i> tariff.tariffId <i>Test System99</i> tariff.currency <i>EUR</i> tariff.chargingTime.prices[0].priceMinute <i>1.00</i> tariff.chargingTime.taxRates[0].tax 0 tariff.chargingTime.taxRates[0].type <i>VAT</i>
10. The Charging Station sends a TransactionEventRequest	11. The Test System responds with a TransactionEventResponse
12. The Charging Station sends a TransactionEventRequest	13. The Test System responds with a TransactionEventResponse

Main (Test scenario)	
<u>Note:</u> Wait 1 minute. Steps 12 and 13 will be repeated every 15 seconds	
14. The Charging Station sends a TransactionEventRequest	15. The Test System responds with a TransactionEventResponse

Tool validations
<p>* Step 2: Message SetDefaultTariffResponse - status must be <i>Accepted</i></p> <p>* Step 4: Message: TransactionEventRequest - triggerReason is <i>RunningCost</i> - costDetails is <i><not omitted></i></p> <p>* Step 6: Message: TransactionEventRequest - triggerReason is <i>RunningCost</i> - costDetails.totalCost.chargingTime.exclTax is between <i>0.10</i> and <i>0.20</i> - and/or costDetails.totalCost.chargingTime.inclTax is between <i>0.10</i> and <i>0.20</i> - costDetails.chargingPeriods must be <i><omitted></i></p> <p>* Step 9: Message ChangeTransactionTariffResponse - status must be <i>Accepted</i></p> <p>* Step 10: Message: TransactionEventRequest - triggerReason must be <i>TariffChanged</i> - transactionInfo.tariffId must be <i>Test System99</i></p> <p>* Step 12: Message: TransactionEventRequest - triggerReason is <i>RunningCost</i> - costDetails is <i><not omitted></i></p> <p>* Step 14: Message: TransactionEventRequest - triggerReason is <i>RunningCost</i> - costDetails.totalCost.currency must be <i>EUR</i> - costDetails.totalCost.chargingTime.exclTax must be between <i>1.00</i> and <i>2.00</i> - costDetails.totalCost.chargingTime.inclTax must be between <i>1.00</i> and <i>2.00</i> - costDetails.chargingPeriods must be <i><omitted></i></p> <p>Post scenario validations: N/a</p>

TC_I_117_CS: Local Cost Calculation - Change transaction tariff - validations

Test case name	Local Cost Calculation - Change transaction tariff - validations
Test case Id	TC_I_117_CS
Use case Id(s)	I11
Requirement(s)	I11.FR.04, I11.FR.05
System under test	Charging Station
Description	CSMS changes the tariff that is associated with a transaction. This may be needed when dealing with unexpected price changes.
Purpose	To verify if the Charging Station correctly validates the tariff change.
Prerequisite(s)	- The Charging Station supports local cost calculation using the Tariff mechanism (TariffCostCtrlr.Enabled[Tariff] exists and <i>ReadWrite</i> or is <i>true</i> and <i>ReadOnly</i>).

Before (Preparations)
Configuration State: - TariffCostCtrlr.Enabled[Tariff] is <i>true</i>
Memory State: N/a
Reusable State: State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with ChangeTransactionTariffResponse	1. The Test System sends a ChangeTransactionTariffRequest with transactionId <unknown transactionId> tariff.tariffId Test System99 tariff.currency EUR tariff.chargingTime.prices[0].priceMinute 6.00
4. The Charging Station responds with ChangeTransactionTariffResponse	3. The Test System sends a ChangeTransactionTariffRequest with transactionId <transaction id> tariff.tariffId Test System1 tariff.currency EUR tariff.chargingTime.prices[0].priceMinute 6.00
6. The Charging Station responds with ChangeTransactionTariffResponse	5. The Test System sends a ChangeTransactionTariffRequest with transactionId <transaction id> tariff.tariffId Test System2 tariff.currency ALL tariff.chargingTime.prices[0].priceMinute 6.00

Tool validations
* Step 2: Message ChangeTransactionTariffResponse - status must be <i>TxNotFound</i>
* Step 4: Message ChangeTransactionTariffResponse - status must be <i>Accepted</i>
* Step 6: Message ChangeTransactionTariffResponse - status must be <i>NoCurrencyChange</i>
Post scenario validations: N/a

TC_I_118_CS: Cost Details of Transaction - no tariff conditions

Test case name	Local Cost Calculation - Cost Details of Transaction - no tariff conditions
Test case Id	TC_I_118_CS
Use case Id(s)	I12
Requirement(s)	I12.FR.01, I12.FR.03, I12.FR.05, I12.FR.09, I12.FR.10, I12.FR.11, I12.FR.12, I12.FR.13, I12.FR.16, I12.FR.17, I12.FR.34
System under test	Charging Station
Description	Charging Station calculates cost of the transaction locally and returns a break-down of the cost at end of the transaction for every charging period for which a different tariff element was active. CSMS can use this to generate invoices or Charge Detail Records for EMSPs.
Purpose	To verify if the Charging Station correctly reports the cost and usage details of a transaction.
Prerequisite(s)	- The Charging Station supports local cost calculation using the Tariff mechanism (TariffCostCtrlr.Enabled[Tariff] exists and <i>ReadWrite</i> or is <i>true</i> and <i>ReadOnly</i>).

Before (Preparations)
Configuration State: - TariffCostCtrlr.Enabled[Tariff] is <i>true</i> - TariffCostCtrlr.Enabled[RunningCost] is <i>true</i> - TariffCostCtrlr.Enabled[Cost] is <i>true</i> - TariffCostCtrlr.Interval[Cost] is <i>15</i> - SampledDataCtrlr.TxUpdatedInterval is <i>0</i> - AlignedDataCtrlr.Interval is <i>0</i>
Memory State: N/a
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS

Main (Test scenario)	
2. The Charging Station responds with SetDefaultTariffResponse	<p>1. The Test System sends a SetDefaultTariffRequest with evseld 0</p> <p>tariff.tariffId <i>Test System1</i></p> <p>tariff.currency <i>EUR</i></p> <p>tariff.energy.prices[0].priceKwh <i>5.00</i></p> <p>tariff.energy.taxRates[0].type <i>t4_5</i></p> <p>tariff.energy.taxRates[0].tax <i>4.5</i></p> <p>tariff.energy.taxRates[1].type <i>t5_5</i></p> <p>tariff.energy.taxRates[1].tax <i>5.5</i></p> <p>tariff.chargingTime.prices[0].priceMinute <i>6.00</i></p> <p>tariff.chargingTime.prices[1].priceMinute <i>60.00</i></p> <p>tariff.chargingTime.taxRates[0].type <i>t34_5</i></p> <p>tariff.chargingTime.taxRates[0].tax <i>34.5</i></p> <p>tariff.chargingTime.taxRates[1].type <i>t5_5</i></p> <p>tariff.chargingTime.taxRates[1].tax <i>5.5</i></p> <p>tariff.idleTime.prices[0].priceMinute <i>12.00</i></p> <p>tariff.idleTime.taxRates[0].type <i>t24_5</i></p> <p>tariff.idleTime.taxRates[0].tax <i>24.5</i></p> <p>tariff.idleTime.taxRates[1].type <i>t5_5</i></p> <p>tariff.idleTime.taxRates[1].tax <i>5.5</i></p> <p>tariff.fixedFee.prices[0].priceFixed <i>10.00</i></p> <p>tariff.fixedFee.taxRates[0].type <i>t14_5</i></p> <p>tariff.fixedFee.taxRates[0].tax <i>14.5</i></p> <p>tariff.fixedFee.taxRates[1].type <i>t5_5</i></p> <p>tariff.fixedFee.taxRates[1].tax <i>5.5</i></p>
3. Execute Reusable State <i>EnergyTransferStarted</i>	
<u>Note:</u> After approx. 10-15 seconds (depending on TxStartPoint) first CostDetails are sent	
4. The Charging Station sends a TransactionEventRequest	5. The Test System responds with a TransactionEventResponse
6. The Charging Station sends a TransactionEventRequest	7. The Test System responds with a TransactionEventResponse
8. The Charging Station sends a TransactionEventRequest	9. The Test System responds with a TransactionEventResponse
10. Execute Reusable State <i>StopAuthorized</i> Execute Reusable State <i>EVDIsconnected</i> Execute Reusable State <i>ParkingBayUnoccupied</i>	

Tool validations

* Step 4: (t = ~15s)

Message **TransactionEventRequest**

- **eventType** must be *Updated*
- **triggerReason** must be *RunningCost*
- **costDetails.failureToCalculate** must be *<omitted>*
- **costDetails.failureReason** must be *<omitted>*
- **costDetails.totalCost.currency** must be *EUR*
- **costDetails.totalCost.typeOfCost** must be *NormalCost*
- **costDetails.totalCost.reservationTime** must be *<omitted>*
- **costDetails.totalCost.reservationFixed** must be *<omitted>*

- **costDetails.totalCost.energy.exclTax** must be *<not omitted>*
- **costDetails.totalCost.energy.inclTax** must be *<costDetails.totalCost.energy.exclTax * 1.1>*
- **costDetails.totalCost.energy.taxRates[0].type** *t4_5*
- **costDetails.totalCost.energy.taxRates[0].tax** *4.5*
- **costDetails.totalCost.energy.taxRates[1].type** *t5_5*
- **costDetails.totalCost.energy.taxRates[1].tax** *5.5*
- **costDetails.totalUsage.energy** must be *> 0.00*

- **costDetails.totalUsage.chargingTime** must be *15*
- **costDetails.totalCost.chargingTime.exclTax** must be *1.50*
- **costDetails.totalCost.chargingTime.inclTax** must be *<costDetails.totalCost.chargingTime.exclTax * 1.4>*
- **costDetails.totalCost.chargingTime.taxRates[0].type** *t34_5*
- **costDetails.totalCost.chargingTime.taxRates[0].tax** *34.5*
- **costDetails.totalCost.chargingTime.taxRates[1].type** *t5_5*
- **costDetails.totalCost.chargingTime.taxRates[1].tax** *5.5*

- **costDetails.totalUsage.idleTime** must be *0*
- **costDetails.totalCost.idleTime.exclTax** must be *0*
- **costDetails.totalCost.idleTime.inclTax** must be *<costDetails.totalCost.idleTime.exclTax * 1.3>*
- **costDetails.totalCost.idleTime.taxRates[0].type** *t24_5*
- **costDetails.totalCost.idleTime.taxRates[0].tax** *24.5*
- **costDetails.totalCost.idleTime.taxRates[1].type** *t5_5*
- **costDetails.totalCost.idleTime.taxRates[1].tax** *5.5*

- **costDetails.totalCost.fixed.exclTax** must be *10*
- **costDetails.totalCost.fixed.inclTax** must be *<costDetails.totalCost.fixed.exclTax * 1.2>*
- **costDetails.totalCost.fixed.taxRates[0].type** *t14_5*
- **costDetails.totalCost.fixed.taxRates[0].tax** *14.5*
- **costDetails.totalCost.fixed.taxRates[1].type** *t5_5*
- **costDetails.totalCost.fixed.taxRates[1].tax** *5.5*

- **costDetails.chargingPeriods** must be *<omitted>*

Tool validations

* Step 6: (t = ~30s)

Message **TransactionEventRequest**

- **eventType** must be *Updated*
- **triggerReason** must be *RunningCost*
- **costDetails.failureToCalculate** must be *<omitted>*
- **costDetails.failureReason** must be *<omitted>*
- **costDetails.totalCost.currency** must be *EUR*
- **costDetails.totalCost.typeOfCost** must be *NormalCost*
- **costDetails.totalCost.reservationTime** must be *<omitted>*
- **costDetails.totalCost.reservationFixed** must be *<omitted>*

- **costDetails.totalCost.energy.exclTax** must be *<not omitted>*
- **costDetails.totalCost.energy.inclTax** must be *<costDetails.totalCost.energy.exclTax * 1.1>*
- **costDetails.totalCost.energy.taxRates[0].type** *t4_5*
- **costDetails.totalCost.energy.taxRates[0].tax** *4.5*
- **costDetails.totalCost.energy.taxRates[1].type** *t5_5*
- **costDetails.totalCost.energy.taxRates[1].tax** *5.5*
- **costDetails.totalUsage.energy** must be *> 0.00*

- **costDetails.totalUsage.chargingTime** must be *30*
- **costDetails.totalCost.chargingTime.exclTax** must be *3.00*
- **costDetails.totalCost.chargingTime.inclTax** must be *<costDetails.totalCost.chargingTime.exclTax * 1.4>*
- **costDetails.totalCost.chargingTime.taxRates[0].type** *t34_5*
- **costDetails.totalCost.chargingTime.taxRates[0].tax** *34.5*
- **costDetails.totalCost.chargingTime.taxRates[1].type** *t5_5*
- **costDetails.totalCost.chargingTime.taxRates[1].tax** *5.5*

- **costDetails.totalUsage.idleTime** must be *0*
- **costDetails.totalCost.idleTime.exclTax** must be *0*
- **costDetails.totalCost.idleTime.inclTax** must be *<costDetails.totalCost.idleTime.exclTax * 1.3>*
- **costDetails.totalCost.idleTime.taxRates[0].type** *t24_5*
- **costDetails.totalCost.idleTime.taxRates[0].tax** *24.5*
- **costDetails.totalCost.idleTime.taxRates[1].type** *t5_5*
- **costDetails.totalCost.idleTime.taxRates[1].tax** *5.5*

- **costDetails.totalCost.fixed.exclTax** must be *10*
- **costDetails.totalCost.fixed.inclTax** must be *<costDetails.totalCost.fixed.exclTax * 1.2>*
- **costDetails.totalCost.fixed.taxRates[0].type** *t14_5*
- **costDetails.totalCost.fixed.taxRates[0].tax** *14.5*
- **costDetails.totalCost.fixed.taxRates[1].type** *t5_5*
- **costDetails.totalCost.fixed.taxRates[1].tax** *5.5*

- **costDetails.chargingPeriods** must be *<omitted>*

Tool validations

* Step 8: (t = ~45s)

Message **TransactionEventRequest**

- **eventType** must be *Updated*
- **triggerReason** must be *RunningCost*
- **costDetails.failureToCalculate** must be *<omitted>*
- **costDetails.failureReason** must be *<omitted>*
- **costDetails.totalCost.currency** must be *EUR*
- **costDetails.totalCost.typeOfCost** must be *NormalCost*
- **costDetails.totalCost.reservationTime** must be *<omitted>*
- **costDetails.totalCost.reservationFixed** must be *<omitted>*

- **costDetails.totalCost.energy.exclTax** must be *<not omitted>*
- **costDetails.totalCost.energy.inclTax** must be *<costDetails.totalCost.energy.exclTax * 1.1>*
- **costDetails.totalCost.energy.taxRates[0].type** *t4_5*
- **costDetails.totalCost.energy.taxRates[0].tax** *4.5*
- **costDetails.totalCost.energy.taxRates[1].type** *t5_5*
- **costDetails.totalCost.energy.taxRates[1].tax** *5.5*
- **costDetails.totalUsage.energy** must be *> 0.00*

- **costDetails.totalUsage.chargingTime** must be *45*
- **costDetails.totalCost.chargingTime.exclTax** must be *4.50*
- **costDetails.totalCost.chargingTime.inclTax** must be *<costDetails.totalCost.chargingTime.exclTax * 1.4>*
- **costDetails.totalCost.chargingTime.taxRates[0].type** *t34_5*
- **costDetails.totalCost.chargingTime.taxRates[0].tax** *34.5*
- **costDetails.totalCost.chargingTime.taxRates[1].type** *t5_5*
- **costDetails.totalCost.chargingTime.taxRates[1].tax** *5.5*

- **costDetails.totalUsage.idleTime** must be *0*
- **costDetails.totalCost.idleTime.exclTax** must be *0*
- **costDetails.totalCost.idleTime.inclTax** must be *<costDetails.totalCost.idleTime.exclTax * 1.3>*
- **costDetails.totalCost.idleTime.taxRates[0].type** *t24_5*
- **costDetails.totalCost.idleTime.taxRates[0].tax** *24.5*
- **costDetails.totalCost.idleTime.taxRates[1].type** *t5_5*
- **costDetails.totalCost.idleTime.taxRates[1].tax** *5.5*

- **costDetails.totalCost.fixed.exclTax** must be *10*
- **costDetails.totalCost.fixed.inclTax** must be *<costDetails.totalCost.fixed.exclTax * 1.2>*
- **costDetails.totalCost.fixed.taxRates[0].type** *t14_5*
- **costDetails.totalCost.fixed.taxRates[0].tax** *14.5*
- **costDetails.totalCost.fixed.taxRates[1].type** *t5_5*
- **costDetails.totalCost.fixed.taxRates[1].tax** *5.5*

- **costDetails.chargingPeriods** must be *<omitted>*

Tool validations**Post scenario validations:**

Note: The value of **TransactionEventRequest.timestamp** of step 3 where **TransactionEventRequest.eventType** is Started is called **<TxStartDateTime>**.

The value of **TransactionEventRequest.timestamp** of step 10 where **TransactionEventRequest.eventType** is Ended is called **<TxEndDateTime>**:

* A message **TransactionEventRequest** must have been received with:

- **eventType** must be *Ended*
- **costDetails.failureToCalculate** must be *<omitted>*
- **costDetails.failureReason** must be *<omitted>*
- **costDetails.totalCost.currency** must be *EUR*
- **costDetails.totalCost.typeOfCost** must be *NormalCost*
- **costDetails.totalCost.reservationTime** must be *<omitted>*
- **costDetails.totalCost.reservationFixed** must be *<omitted>*

- **costDetails.totalCost.energy.exclTax** must be *<not omitted>*
- **costDetails.totalCost.energy.inclTax** must be *<costDetails.totalCost.energy.exclTax * 1.1>*
- **costDetails.totalCost.energy.taxRates[0].type** *t4_5*
- **costDetails.totalCost.energy.taxRates[0].tax** *4.5*
- **costDetails.totalCost.energy.taxRates[1].type** *t5_5*
- **costDetails.totalCost.energy.taxRates[1].tax** *5.5*
- **costDetails.totalUsage.energy** must be *> 0.00*

- **costDetails.totalUsage.chargingTime** must be *<TxEndDateTime - TxStartDateTime>*
- **costDetails.totalCost.chargingTime.exclTax** must be *chargingTime * 0.1 EUR/s*
- **costDetails.totalCost.chargingTime.inclTax** must be *<costDetails.totalCost.chargingTime.exclTax * 1.4>*
- **costDetails.totalCost.chargingTime.taxRates[0].type** *t34_5*
- **costDetails.totalCost.chargingTime.taxRates[0].tax** *34.5*
- **costDetails.totalCost.chargingTime.taxRates[1].type** *t5_5*
- **costDetails.totalCost.chargingTime.taxRates[1].tax** *5.5*

Tool validations

- **costDetails.totalCost.idleTime.exclTax** must be 0
- **costDetails.totalUsage.idleTime** must be 0
- **costDetails.totalCost.idleTime.inclTax** must be $\langle \text{costDetails.totalCost.idleTime.exclTax} * 1.3 \rangle$
- **costDetails.totalCost.idleTime.taxRates[0].type** t24_5
- **costDetails.totalCost.idleTime.taxRates[0].tax** 24.5
- **costDetails.totalCost.idleTime.taxRates[1].type** t5_5
- **costDetails.totalCost.idleTime.taxRates[1].tax** 5.5

- **costDetails.totalCost.fixed.exclTax** must be 10
- **costDetails.totalCost.fixed.inclTax** must be $\langle \text{costDetails.totalCost.fixed.exclTax} * 1.2 \rangle$
- **costDetails.totalCost.fixed.taxRates[0].type** t14_5
- **costDetails.totalCost.fixed.taxRates[0].tax** 14.5
- **costDetails.totalCost.fixed.taxRates[1].type** t5_5
- **costDetails.totalCost.fixed.taxRates[1].tax** 5.5

- **costDetails.chargingPeriods[0].tariffId** must be *Test System1*
- **costDetails.chargingPeriods[0].startPeriod** = $\langle \text{TxStartDateTime} \rangle$
- **costDetails.chargingPeriods[0].dimensions[0].type** must be *Energy*
- **costDetails.chargingPeriods[0].dimensions[0].volume** must be $\langle > 0 \rangle$
- **costDetails.chargingPeriods[0].dimensions[1].type** must be *ChargingTime*
- **costDetails.chargingPeriods[0].dimensions[1].volume** must be $\langle \text{TxEndDateTime} - \text{TxStartDateTime} \rangle$
- **costDetails.chargingPeriods[0].dimensions[2].type** must be *_IdleTime*
- **costDetails.chargingPeriods[0].dimensions[2].volume** must be 0

TC_I_119_CS: Local Cost Calculation - Cost Details of Transaction - with tariff conditions

Test case name	Local Cost Calculation - Cost Details of Transaction - with tariff conditions
Test case Id	TC_I_119_CS
Use case Id(s)	I12
Requirement(s)	I12.FR.06, I12.FR.30, I12.FR.35, I12.FR.36
System under test	Charging Station
Description	Charging Station calculates cost of the transaction locally and returns a break-down of the cost at end of the transaction for every charging period for which a different tariff element was active. CSMS can use this to generate invoices or Charge Detail Records for EMSPs.
Purpose	To verify if the Charging Station correctly reports the cost and usage details of a transaction.
Prerequisite(s)	- The Charging Station supports local cost calculation using the Tariff mechanism (TariffCostCtrlr.Enabled[Tariff] exists and <i>ReadWrite</i> or is <i>true</i> and <i>ReadOnly</i>). - The Charging Station supports tariff conditions (TariffCostCtrlr.ConditionsSupported[Tariff] is <i>true</i>)

Before (Preparations)
Configuration State: - TariffCostCtrlr.Enabled[Tariff] is <i>true</i> - TariffCostCtrlr.Enabled[RunningCost] is <i>true</i> - TariffCostCtrlr.Enabled[Cost] is <i>true</i> - TariffCostCtrlr.Interval[Cost] is 15 - SampledDataCtrlr.TxUpdatedInterval is 0 - AlignedDataCtrlr.Interval is 0
Memory State: N/a
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with SetDefaultTariffResponse	<p>1. The Test System sends a SetDefaultTariffRequest with evseld 0 tariff.tariffId <i>Test System1</i> tariff.currency <i>EUR</i> tariff.reservationFixed.prices[0].priceFixed <i>20.00</i> tariff.reservationFixed.taxRates[0].type <i>t5_0</i> tariff.reservationFixed.taxRates[0].tax <i>5.0</i></p> <p>tariff.fixedFee.prices[0].priceFixed <i>10.00</i> tariff.fixedFee.prices[2].priceFixed <i>5.00</i> tariff.fixedFee.taxRates[0].type <i>t5_0</i> tariff.fixedFee.taxRates[0].tax <i>5.0</i></p> <p>tariff.chargingTime.prices[0].priceMinute <i>6.00</i> tariff.chargingTime.prices[0].conditions.minTime <i>30</i> tariff.chargingTime.prices[1].priceMinute <i>60.00</i> tariff.chargingTime.taxRates[0].type <i>t20_0</i> tariff.chargingTime.taxRates[0].tax <i>20.0</i></p>
3. Execute Reusable State <i>EnergyTransferStarted</i>	
<u>Note:</u> After approx 10-15 s (depending on TxStartPoint) first CostDetails are sent.	

Main (Test scenario)	
4. The Charging Station sends a TransactionEventRequest	5. The Test System responds with a TransactionEventResponse
6. The Charging Station sends a TransactionEventRequest	7. The Test System responds with a TransactionEventResponse
8. The Charging Station sends a TransactionEventRequest	9. The Test System responds with a TransactionEventResponse
10. Execute Reusable State <i>StopAuthorized</i> Execute Reusable State <i>EVDDisconnected</i> Execute Reusable State <i>ParkingBayUnoccupied</i>	

Tool validations
<p>* Step 4: (t = ~15s)</p> <p>Message TransactionEventRequest</p> <ul style="list-style-type: none"> - eventType must be <i>Updated</i> - triggerReason must be <i>RunningCost</i> - costDetails.failureToCalculate must be <i><omitted></i> - costDetails.failureReason must be <i><omitted></i> - costDetails.totalCost.currency must be <i>EUR</i> - costDetails.totalCost.typeOfCost must be <i>NormalCost</i> - costDetails.totalCost.reservationTime must be <i><omitted></i> - costDetails.totalCost.reservationFixed must be <i><omitted></i> <ul style="list-style-type: none"> - costDetails.totalCost.fixed.exclTax must be <i>10</i> - costDetails.totalCost.fixed.inclTax must be <i><costDetails.totalCost.fixed.exclTax * 1.05></i> - costDetails.totalCost.fixed.taxRates[0].type <i>t5_0</i> - costDetails.totalCost.fixed.taxRates[0].tax <i>5.0</i> <ul style="list-style-type: none"> - costDetails.totalUsage.chargingTime must be <i>15</i> - costDetails.totalCost.chargingTime.exclTax must be <i>15</i> - costDetails.totalCost.chargingTime.inclTax must be <i><costDetails.totalCost.chargingTime.exclTax * 1.2></i> - costDetails.totalCost.chargingTime.taxRates[0].type <i>t20</i> - costDetails.totalCost.chargingTime.taxRates[0].tax <i>20</i> <ul style="list-style-type: none"> - costDetails.chargingPeriods must be <i><omitted></i>

Tool validations

* Step 6: (t = ~30s)

Message **TransactionEventRequest**

- **eventType** must be *Updated*
- **triggerReason** must be *RunningCost*
- **costDetails.failureToCalculate** must be *<omitted>*
- **costDetails.failureReason** must be *<omitted>*
- **costDetails.totalCost.currency** must be *EUR*
- **costDetails.totalCost.typeOfCost** must be *NormalCost*
- **costDetails.totalCost.reservationTime** must be *<omitted>*
- **costDetails.totalCost.reservationFixed** must be *<omitted>*

- **costDetails.totalCost.fixed.exclTax** must be *10*
- **costDetails.totalCost.fixed.inclTax** must be *<costDetails.totalCost.fixed.exclTax * 1.05>*
- **costDetails.totalCost.fixed.taxRates[0].type** *t5_0*
- **costDetails.totalCost.fixed.taxRates[0].tax** *5.0*

- **costDetails.totalUsage.chargingTime** must be *30*
- **costDetails.totalCost.chargingTime.exclTax** must be *30*
- **costDetails.totalCost.chargingTime.inclTax** must be *<costDetails.totalCost.chargingTime.exclTax * 1.2>*
- **costDetails.totalCost.chargingTime.taxRates[0].type** *t20*
- **costDetails.totalCost.chargingTime.taxRates[0].tax** *20*

- **costDetails.chargingPeriods** must be *<omitted>*

* Step 8: (t = ~45s)

Message **TransactionEventRequest**

- **eventType** must be *Updated*
- **triggerReason** must be *RunningCost*
- **costDetails.failureToCalculate** must be *<omitted>*
- **costDetails.failureReason** must be *<omitted>*
- **costDetails.totalCost.currency** must be *EUR*
- **costDetails.totalCost.typeOfCost** must be *NormalCost*
- **costDetails.totalCost.reservationTime** must be *<omitted>*
- **costDetails.totalCost.reservationFixed** must be *<omitted>*

- **costDetails.totalCost.fixed.exclTax** must be *10*
- **costDetails.totalCost.fixed.inclTax** must be *<costDetails.totalCost.fixed.exclTax * 1.05>*
- **costDetails.totalCost.fixed.taxRates[0].type** *t5_0*
- **costDetails.totalCost.fixed.taxRates[0].tax** *5.0*

- **costDetails.totalUsage.chargingTime** must be *45*
- **costDetails.totalCost.chargingTime.exclTax** must be *31.50*
- **costDetails.totalCost.chargingTime.inclTax** must be *<costDetails.totalCost.chargingTime.exclTax * 1.2>*
- **costDetails.totalCost.chargingTime.taxRates[0].type** *t20*
- **costDetails.totalCost.chargingTime.taxRates[0].tax** *20*

Tool validations

Post scenario validations:

Note: The value of **TransactionEventRequest.timestamp** of step 3 where **TransactionEventRequest.eventType** is Started is called **<TxStartDateTime>**.

The value of **TransactionEventRequest.timestamp** of step 10 where **TransactionEventRequest.eventType** is Ended is called **<TxEndDateTime>**:

* A message **TransactionEventRequest** must have been received with:

- **eventType** must be *Ended*
- **costDetails.failureToCalculate** must be *<omitted>*
- **costDetails.failureReason** must be *<omitted>*
- **costDetails.totalCost.currency** must be *EUR*
- **costDetails.totalCost.typeOfCost** must be *NormalCost*
- **costDetails.totalCost.reservationTime** must be *<omitted>*
- **costDetails.totalCost.reservationFixed** must be *<omitted>*

- **costDetails.totalCost.fixed.exclTax** must be *10*
- **costDetails.totalCost.fixed.inclTax** must be *<costDetails.totalCost.fixed.exclTax * 1.05>*
- **costDetails.totalCost.fixed.taxRates[0].type** *t5_0*
- **costDetails.totalCost.fixed.taxRates[0].tax** *5.0*

- **costDetails.totalUsage.chargingTime** must be *<TxEndDateTime> - <TxStartDateTime>*
- **costDetails.totalCost.chargingTime.exclTax** must be *30 * 1 EUR/s + (chargingTime - 30) * 0.1 EUR/s*
- **costDetails.totalCost.chargingTime.inclTax** must be *<costDetails.totalCost.chargingTime.exclTax * 1.2>*
- **costDetails.totalCost.chargingTime.taxRates[0].type** *t20*
- **costDetails.totalCost.chargingTime.taxRates[0].tax** *20*

- **costDetails.chargingPeriods[0].tariffId** must be *Test System1*
- **costDetails.chargingPeriods[0].startPeriod** must be *<TxStartDateTime>*
- **costDetails.chargingPeriods[0].dimensions[0].type** must be *ChargingTime*
- **costDetails.chargingPeriods[0].dimensions[0].volume** must be *30*
- **costDetails.chargingPeriods[1].tariffId** must be *Test System1*
- **costDetails.chargingPeriods[1].startPeriod** must be *<txStartDateTime> + 30*
- **costDetails.chargingPeriods[1].dimensions[0].type** must be *ChargingTime*
- **costDetails.chargingPeriods[1].dimensions[0].volume** must be *(<TxEndDateTime> - <TxStartDateTime> - 30)*

TC_I_120_CS: Local Cost Calculation - Cost Details of Transaction - reservation

Test case name	Local Cost Calculation - Cost Details of Transaction - reservation
Test case Id	TC_I_120_CS
Use case Id(s)	I12
Requirement(s)	I12.FR.06, I12.FR.07, I12.FR.08
System under test	Charging Station
Description	Charging Station calculates cost of the transaction locally and returns a break-down of the cost at end of the transaction for every charging period for which a different tariff element was active. CSMS can use this to generate invoices or Charge Detail Records for EMSPs.
Purpose	To verify if the Charging Station is able to calculate a fixed price component next to a fixed price and a time price for reservations.
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station supports local cost calculation using the Tariff mechanism (TariffCostCtrlr.Enabled[Tariff] exists and <i>ReadWrite</i> or is <i>true</i> and <i>ReadOnly</i>). - The Charging Station supports tariff conditions (TariffCostCtrlr.ConditionsSupported[Tariff] is <i>true</i>)

Before (Preparations)

Configuration State:

- **TariffCostCtrlr.Enabled[Tariff]** is *true*
- **TariffCostCtrlr.Enabled[RunningCost]** is *true*
- **TariffCostCtrlr.Enabled[Cost]** is *true*
- **TariffCostCtrlr.Interval[Cost]** is 15
- **SampledDataCtrlr.TxUpdatedInterval** is 0
- **AlignedDataCtrlr.Interval** is 0

Memory State:

N/a

Reusable State(s):

N/a

Main (Test scenario)

Charging Station	CSMS
2. The Charging Station responds with SetDefaultTariffResponse	<p>1. The Test System sends a SetDefaultTariffRequest with evseld 0 tariff.tariffId <i>Test System1</i> tariff.currency <i>EUR</i></p> <p>tariff.fixedFee.prices[0].priceFixed 10.00 tariff.fixedFee.taxRates[0].type <i>t5_0</i> tariff.fixedFee.taxRates[0].tax 5.0</p> <p>tariff.reservationFixed.prices[0].priceFixed 20.00 tariff.reservationFixed.taxRates[0].type <i>t5_0</i> tariff.reservationFixed.taxRates[0].tax 5.0</p> <p>tariff.reservationTime.prices[0].priceMinute 6.00 tariff.reservationTime.taxRates[0].type <i>t20_0</i> tariff.reservationTime.taxRates[0].tax 20.0</p>
4. The Charging Station responds with a ReserveNowResponse	<p>3. The Test System sends a ReserveNowRequest with evseld <omitted> idToken.idToken <Configured valid_idtoken_idtoken> idToken.type <Configured valid_idtoken_type></p>

Main (Test scenario)	
<u>Note:</u> The Test System waits 30 seconds	
5. Execute Reusable State <i>EnergyTransferStarted</i>	
<u>Note:</u> After ~10 - 15s (depending on TxStartPoint)	
6. The Charging Station sends a TransactionEventRequest	7. The Test System responds with a TransactionEventResponse
8. Execute Reusable State <i>StopAuthorized</i> Execute Reusable State <i>EVDisconnected</i> Execute Reusable State <i>ParkingBayUnoccupied</i>	

Tool validations
<p>* Step 4: Message ReserveNowResponse - status must be <i>Accepted</i> <u>Note:</u> Time of reservation response is referred to as <ReservationTime>.</p> <p>* Step 6: <u>Note:</u> The value of TransactionEventRequest.timestamp of step 6 where TransactionEventRequest.eventType is <i>Started</i> is called <TxStartDateTime>.</p> <p>Message TransactionEventRequest - eventType must be <i>Updated</i> - triggerReason must be <i>RunningCost</i> - costDetails.failureToCalculate must be <omitted> - costDetails.failureReason must be <omitted> - costDetails.totalCost.currency must be <i>EUR</i> - costDetails.totalCost.typeOfCost must be <i>NormalCost</i> - costDetails.totalCost.fixed.exclTax must be <i>10.00</i> - costDetails.totalCost.fixed.inclTax must be $10.00 * 1.05$ - costDetails.totalCost.fixed.taxRates[0].type <i>t5_0</i> - costDetails.totalCost.fixed.taxRates[0].tax <i>5.0</i> - costDetails.totalCost.reservationFixed.exclTax must be <i>20.00</i> - costDetails.totalCost.reservationFixed.inclTax must be $20.00 * 1.05$ - costDetails.totalCost.reservationFixed.taxRates[0].type <i>t5_0</i> - costDetails.totalCost.reservationFixed.taxRates[0].tax <i>5.0</i> - costDetails.totalCost.reservationTime.exclTax must be $(\text{TxStartDateTime} - \text{ReservationTime}) / 60 * 6.00$ - costDetails.totalCost.reservationTime.inclTax must be $(\text{TxStartDateTime} - \text{ReservationTime}) / 60 * 6.00 * 1.20$ - costDetails.totalCost.reservationTime.taxRates[0].type <i>t20_0</i> - costDetails.totalCost.reservationTime.taxRates[0].tax <i>20.0</i> - costDetails.totalUsage.energy must be <not omitted> - costDetails.totalUsage.chargingTime must be <not omitted> - costDetails.totalUsage.idleTime must be <not omitted> - costDetails.totalUsage.reservationTime must be $(\text{TxStartDateTime} - \text{ReservationTime}) / 60$ - costDetails.chargingPeriods must be <omitted></p>

Tool validations

Post scenario validations:

Note: The value of **TransactionEventRequest.timestamp** of step 6 where **TransactionEventRequest.eventType** is Started is called **<TxStartDateTime>**.

* A message **TransactionEventRequest** must have been received with:

- **eventType** must be *Ended*
- **costDetails.failureToCalculate** must be *<omitted>*
- **costDetails.failureReason** must be *<omitted>*
- **costDetails.totalCost.currency** must be *EUR*
- **costDetails.totalCost.typeOfCost** must be *NormalCost*
- **costDetails.totalCost.fixed.exclTax** must be *10.00*
- **costDetails.totalCost.fixed.inclTax** must be *10.00 * 1.05*
- **costDetails.totalCost.fixed.taxRates[0].type** *t5_0*
- **costDetails.totalCost.fixed.taxRates[0].tax** *5.0*
- **costDetails.totalCost.reservationFixed.exclTax** must be *20.00*
- **costDetails.totalCost.reservationFixed.inclTax** must be *20.00 * 1.05*
- **costDetails.totalCost.reservationFixed.taxRates[0].type** *t5_0*
- **costDetails.totalCost.reservationFixed.taxRates[0].tax** *5.0*
- **costDetails.totalCost.reservationTime.exclTax** must be *(<TxStartDateTime> - <ReservationTime>) / 60 * 6.00*
- **costDetails.totalCost.reservationTime.inclTax** must be *(<TxStartDateTime> - <ReservationTime>) / 60 * 6.00 * 1.20*
- **costDetails.totalCost.reservationTime.taxRates[0].type** *t20_0*
- **costDetails.totalCost.reservationTime.taxRates[0].tax** *20.0*
- **costDetails.totalUsage.energy** must be *<not omitted>*
- **costDetails.totalUsage.chargingTime** must be *<not omitted>*
- **costDetails.totalUsage.idleTime** must be *<not omitted>*
- **costDetails.totalUsage.reservationTime** must be *(<TxStartDateTime> - <ReservationTime>) / 60*

- **costDetails.chargingPeriods[0].tariffId** must be *Test System1*
- **costDetails.chargingPeriods[0].startPeriod** must be *<TxStartDateTime>*
- **costDetails.chargingPeriods[0].dimensions[0].type** must be *ChargingTime*
- **costDetails.chargingPeriods[0].dimensions[0].volume** must be *_(<TxEndDateTime> - <TxStartDateTime>)_*

TC_I_121_CS: Local Cost Calculation - Cost Details of Transaction - minCost/maxCost

Test case name	Local Cost Calculation - Cost Details of Transaction - minCost/maxCost
Test case Id	TC_I_121_CS
Use case Id(s)	I12
Requirement(s)	I12.FR.17, I12.FR.38, I12.FR.39, I12.FR.39, I12.FR.40
System under test	Charging Station
Description	Charging Station calculates cost of the transaction locally and returns a break-down of the cost at end of the transaction for every charging period for which a different tariff element was active. CSMS can use this to generate invoices or Charge Detail Records for EMSPs.
Purpose	To verify if the Charging Station correctly reports the cost and usage details of a transaction.
Prerequisite(s)	- The Charging Station supports local cost calculation using the Tariff mechanism (TariffCostCtrlr.Enabled[Tariff] exists and <i>ReadWrite</i> or is <i>true</i> and <i>ReadOnly</i>). - The Charging Station supports tariff conditions (TariffCostCtrlr.ConditionsSupported[Tariff] is <i>true</i>)

Before (Preparations)
Configuration State: - TariffCostCtrlr.Enabled[Tariff] is <i>true</i> - TariffCostCtrlr.Enabled[RunningCost] is <i>true</i> - TariffCostCtrlr.Enabled[Cost] is <i>true</i> - TariffCostCtrlr.Interval[Cost] is 15 - SampledDataCtrlr.TxUpdatedInterval is 0 - AlignedDataCtrlr.Interval is 0
Memory State: N/a
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with SetDefaultTariffResponse	1. The Test System sends a SetDefaultTariffRequest with evseld 0 tariff.tariffId <i>Test System1</i> tariff.currency <i>EUR</i> tariff.minCost.exclTax 16.00 tariff.minCost.inclTax 17.60 tariff.minCost.taxRates[0].type <i>t10_0</i> tariff.minCost.taxRates[0].tax 10.0 tariff.maxCost.exclTax 35.00 tariff.maxCost.inclTax 38.50 tariff.maxCost.taxRates[0].type <i>t10_0</i> tariff.maxCost.taxRates[0].tax 10.0 tariff.chargingTime.prices[0].priceMinute 60.00 tariff.chargingTime.taxRates[0].type <i>t20_0</i> tariff.chargingTime.taxRates[0].tax 20.0
3. Execute Reusable State <i>EnergyTransferStarted</i>	
<u>Note:</u> The Test System waits 15 seconds	
4. Execute Reusable State <i>StopAuthorized</i> Execute Reusable State <i>EVDIsconnected</i> Execute Reusable State <i>ParkingBayUnoccupied</i>	
5. The Charging Station sends a TransactionEventRequest	6. The Test System responds with a TransactionEventResponse

Main (Test scenario)	
7. Execute Reusable State <i>EnergyTransferStarted</i>	
<u>Note:</u> The Test System waits 30 seconds	
8. Execute Reusable State <i>StopAuthorized</i> Execute Reusable State <i>EVDDisconnected</i> Execute Reusable State <i>ParkingBayUnoccupied</i>	
9. The Charging Station sends a TransactionEventRequest	10. The Test System responds with a TransactionEventResponse
11. Execute Reusable State <i>EnergyTransferStarted</i>	
12. The Charging Station sends a TransactionEventRequest	13. The Test System responds with a TransactionEventResponse
14. The Charging Station sends a TransactionEventRequest	15. The Test System responds with a TransactionEventResponse
16. The Charging Station sends a TransactionEventRequest	17. The Test System responds with a TransactionEventResponse
18. The Charging Station sends a TransactionEventRequest	19. The Test System responds with a TransactionEventResponse
20. Execute Reusable State <i>StopAuthorized</i> Execute Reusable State <i>EVDDisconnected</i> Execute Reusable State <i>ParkingBayUnoccupied</i>	
21. The Charging Station sends a TransactionEventRequest	22. The Test System responds with a TransactionEventResponse

Tool validations
<p>* Step 5: <u>Note:</u> The value of TransactionEventRequest.timestamp of last TransactionEventRequest.eventType is Started is called TxStartDateTime:</p> <p>Message TransactionEventRequest with eventType Ended from reusable states leading up to <i>ParkingBayUnoccupied</i> must have:</p> <ul style="list-style-type: none"> - costDetails.failureToCalculate must be <omitted> - costDetails.failureReason must be <omitted> - costDetails.totalCost.currency must be EUR - costDetails.totalCost.typeOfCost must be MinCost - costDetails.totalCost.fixed.exclTax must be 16.00 - costDetails.totalCost.fixed.inclTax must be 17.60 - costDetails.totalCost.fixed.taxRates[0].type t10_0 - costDetails.totalCost.fixed.taxRates[0].tax 10.0 - costDetails.totalCost.energy must be <omitted> - costDetails.totalCost.chargingTime must be <omitted> - costDetails.totalCost.idleTime must be <omitted> - costDetails.totalCost.reservation must be <omitted> - costDetails.totalCost.total.exclTax must be 16.00 - costDetails.totalCost.total.inclTax must be 17.60 - costDetails.chargingPeriods[0].tariffId must be Test System1 - costDetails.chargingPeriods[0].startPeriod must be <TxStartDateTime> - costDetails.chargingPeriods[0].dimensions[0].type must be ChargingTime - costDetails.chargingPeriods[0].dimensions[0].volume must be <not omitted>

Tool validations

* Step 9:

Note: The value of **TransactionEventRequest.timestamp** of last **TransactionEventRequest.eventType** is Started is called **TxStartDateTime**:

Message **TransactionEventRequest** with **eventType** *Ended* from reusable states leading up to **ParkingBayUnoccupied** must have:

- **costDetails.failureToCalculate** must be <omitted>
- **costDetails.failureReason** must be <omitted>
- **costDetails.totalCost.currency** must be *EUR*
- **costDetails.totalCost.typeOfCost** must be *NormalCost*
- **costDetails.totalCost.chargingTime.exclTax** must be *30.0*
- **costDetails.totalCost.chargingTime.inclTax** must be *36.0*
- **costDetails.totalCost.chargingTime.taxRates[0].type** *t20_0*
- **costDetails.totalCost.chargingTime.taxRates[0].tax** *20.0*
- **costDetails.totalUsage.chargingTime** must be *30*
- **costDetails.chargingPeriods[0].tariffId** must be *Test System1*
- **costDetails.chargingPeriods[0].startPeriod** must be <*TxStartDateTime*>
- **costDetails.chargingPeriods[0].dimensions[0].type** must be *ChargingTime*
- **costDetails.chargingPeriods[0].dimensions[0].volume** must be <not omitted>

Tool validations

* Step 12:

Note: The value of **TransactionEventRequest.timestamp** of last **TransactionEventRequest.eventType** is Started is called TxStartDateTime:

Message **TransactionEventRequest** with **eventType** Ended from reusable states leading up to [ParkingBayUnoccupied](#) must have:

- **costDetails.failureToCalculate** must be <omitted>
- **costDetails.failureReason** must be <omitted>
- **costDetails.totalCost.currency** must be EUR
- **costDetails.totalCost.typeOfCost** must be NormalCost
- **costDetails.totalCost.chargingTime.exclTax** must be 15.00
- **costDetails.totalCost.chargingTime.inclTax** must be 18.50
- **costDetails.totalCost.chargingTime.taxRates[0].type** t20_0
- **costDetails.totalCost.chargingTime.taxRates[0].tax** 20.0
- **costDetails.totalUsage.chargingTime** must be 15
- **costDetails.chargingPeriods[0].tariffId** must be Test System1
- **costDetails.chargingPeriods[0].startPeriod** must be <txStartDateTime>
- **costDetails.chargingPeriods[0].dimensions[0].type** must be ChargingTime
- **costDetails.chargingPeriods[0].dimensions[0].volume** must be 15

* Step 14:

Note: The value of **TransactionEventRequest.timestamp** of last **TransactionEventRequest.eventType** is Started is called TxStartDateTime:

Message **TransactionEventRequest** with **eventType** Ended from reusable states leading up to [ParkingBayUnoccupied](#) must have:

- **costDetails.failureToCalculate** must be <omitted>
- **costDetails.failureReason** must be <omitted>
- **costDetails.totalCost.currency** must be EUR
- **costDetails.totalCost.typeOfCost** must be NormalCost
- **costDetails.totalCost.chargingTime.exclTax** must be 30.00
- **costDetails.totalCost.chargingTime.inclTax** must be 36.00
- **costDetails.totalCost.chargingTime.taxRates[0].type** t20_0
- **costDetails.totalCost.chargingTime.taxRates[0].tax** 20.0
- **costDetails.totalUsage.chargingTime** must be 30
- **costDetails.chargingPeriods[0].tariffId** must be Test System1
- **costDetails.chargingPeriods[0].startPeriod** must be <txStartDateTime>
- **costDetails.chargingPeriods[0].dimensions[0].type** must be ChargingTime
- **costDetails.chargingPeriods[0].dimensions[0].volume** must be 30

Tool validations

* Step 16:

Note: The value of **TransactionEventRequest.timestamp** of last **TransactionEventRequest.eventType** is Started is called TxStartDateTime:

Message **TransactionEventRequest**

- **eventType** must be *Updated*
- **triggerReason** must be *RunningCost*
- **costDetails.failureToCalculate** must be *<omitted>*
- **costDetails.failureReason** must be *<omitted>*
- **costDetails.totalCost.currency** must be *EUR*
- **costDetails.totalCost.typeOfCost** must be *MaxCost*
- **costDetails.totalCost.fixed.exclTax** must be *35.00*
- **costDetails.totalCost.fixed.inclTax** must be *38.50*
- **costDetails.totalCost.fixed.taxRates[0].type** *t10_0*
- **costDetails.totalCost.fixed.taxRates[0].tax** *10.0*
- **costDetails.totalCost.energy** must be *<omitted>*
- **costDetails.totalCost.chargingTime** must be *<omitted>*
- **costDetails.totalCost.idleTime** must be *<omitted>*
- **costDetails.totalCost.reservation** must be *<omitted>*
- **costDetails.totalCost.total.exclTax** must be *35.00*
- **costDetails.totalCost.total.inclTax** must be *38.50*
- **costDetails.chargingPeriods** must be *<omitted>*
- **costDetails.chargingPeriods[0].tariffId** must be *Test System1*
- **costDetails.chargingPeriods[0].startPeriod** must be *<txStartDateTime>*
- **costDetails.chargingPeriods[0].dimensions[0].type** must be *ChargingTime*
- **costDetails.chargingPeriods[0].dimensions[0].volume** must be *45*

* Step 18:

Message **TransactionEventRequest**

- **eventType** must be *Updated*
- **triggerReason** must be *RunningCost*
- **costDetails.failureToCalculate** must be *<omitted>*
- **costDetails.failureReason** must be *<omitted>*
- **costDetails.totalCost.currency** must be *EUR*
- **costDetails.totalCost.typeOfCost** must be *MaxCost*
- **costDetails.totalCost.fixed.exclTax** must be *35.00*
- **costDetails.totalCost.fixed.inclTax** must be *38.50*
- **costDetails.totalCost.fixed.taxRates[0].type** *t10_0*
- **costDetails.totalCost.fixed.taxRates[0].tax** *10.0*
- **costDetails.totalCost.energy** must be *<omitted>*
- **costDetails.totalCost.chargingTime** must be *<omitted>*
- **costDetails.totalCost.idleTime** must be *<omitted>*
- **costDetails.totalCost.reservation** must be *<omitted>*
- **costDetails.totalCost.total.exclTax** must be *35.00*
- **costDetails.totalCost.total.inclTax** must be *38.50*
- **costDetails.chargingPeriods[0].tariffId** must be *Test System1*
- **costDetails.chargingPeriods[0].startPeriod** must be *<txStartDateTime>*
- **costDetails.chargingPeriods[0].dimensions[0].type** must be *ChargingTime*
- **costDetails.chargingPeriods[0].dimensions[0].volume** must be *60*

Tool validations

* Step 21:

Note: The value of **TransactionEventRequest.timestamp** of last **TransactionEventRequest.eventType** is Started is called TxStartDateTime:

Message **TransactionEventRequest** with **eventType** *Ended* from reusable states leading up to **ParkingBayUnoccupied** must have:

- **costDetails.failureToCalculate** must be <omitted>
- **costDetails.failureReason** must be <omitted>
- **costDetails.totalCost.currency** must be *EUR*
- **costDetails.totalCost.typeOfCost** must be *MaxCost*
- **costDetails.totalCost.fixed.exclTax** must be *35.00*
- **costDetails.totalCost.fixed.inclTax** must be *38.50*
- **costDetails.totalCost.fixed.taxRates[0].type** *t10_0*
- **costDetails.totalCost.fixed.taxRates[0].tax** *10.0*
- **costDetails.totalCost.energy** must be <omitted>
- **costDetails.totalCost.chargingTime** must be <omitted>
- **costDetails.totalCost.idleTime** must be <omitted>
- **costDetails.totalCost.reservation** must be <omitted>
- **costDetails.totalCost.total.exclTax** must be *35.00*
- **costDetails.totalCost.total.inclTax** must be *38.50*
- **costDetails.chargingPeriods[0].tariffId** must be *Test System1*
- **costDetails.chargingPeriods[0].startPeriod** must be <txStartDateTime>
- **costDetails.chargingPeriods[0].dimensions[0].type** must be *ChargingTime*
- **costDetails.chargingPeriods[0].dimensions[0].volume** must be > *60*

Post scenario validations:

N/a

J Meter Values

TC_J_01_CS: Clock-aligned Meter Values - No transaction ongoing

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
<p>1. The Charging Station notifies the CSMS about its measured Meter Values.</p> <p>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 Test System will end the testcase after it has received three Meter Value messages.</p>	<p>2. The Test System responds accordingly.</p>

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 (evseld=0). But the timestamp of these messages must all be the same.>
 - None of the provided **sampledValues** shall have **location** = EV, except when **measurand** = SoC.

TC_J_02_CS: Clock-aligned Meter Values - Transaction ongoing

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)

RegisterValuesWithoutPhases is *false* (If implemented)

Memory State:

N/a

Reusable State(s):

State is *EnergyTransferStarted*

Main (Test scenario)

Charging Station	CSMS
<u>Note(s):</u> - 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. <u>Note(s):</u> - During a transaction the <i>MeterValueRequest</i> can still be used to report meter values for the main power meter (<i>evseld=0</i>) 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 Test System responds accordingly.
3. The Charging Station sends a <i>TransactionEventRequest</i> <u>Note(s):</u> - During a transaction the meter values for the configured EVSE with the ongoing transaction should be transmitted using the <i>TransactionEventRequest</i> . - The <i>TransactionEventRequest</i> messages do NOT need to be send/received at the exact specified interval. The configured measurands must be measured at the configured interval. - Multiple <i>TransactionEventRequest</i> messages may be sent per configured interval, in case the amount of measured data is too much for one message. - The Test System will end the testcase after it has the <Configured transaction duration> is reached.	4. The Test System responds with a <i>TransactionEventResponse</i>

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*>

Notes:

- The measurand field may be omitted when the measurand is "Energy.Active.Import.Register"
- It is possible that measurands are reported on multiple locations or phases, based on the capabilities of the energy meter.

* 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*>

Notes:

- The measurand field may be omitted when the measurand is "Energy.Active.Import.Register"
- It is possible that measurands are reported on multiple locations or phases, based on the capabilities of the energy meter.

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.>
- None of the provided **sampledValues** shall have **location** = EV, except when **measurand** = SoC.

TC_J_03_CS: Clock-aligned Meter Values - EventType Ended

Test case name	Clock-aligned Meter Values - EventType Ended
Test case Id	TC_J_03_CS
Use case Id(s)	J01 & (E06,E07,E08,E09,E10,E12)
Requirement(s)	J01.FR.01,J01.FR.02,J01.FR.03,J01.FR.06,J01.FR.07,J01.FR.08,J01.FR.14,J01.FR.15, E06.FR.11,E06.FR.17,E07.FR.08,E07.FR.13,E08.FR.09,E09.FR.05,E10.FR.04,E12.FR.07
System under test	Charging Station
Description	The Charging Station samples the electrical meter or other sensor/transducer hardware to provide information about its Meter Values. Depending on configuration settings, the Charging Station will send Meter Values.
Purpose	To verify if the Charging Station is able to send clock-aligned Meter Values when a transaction ends at the TransactionEventRequest with eventType is <i>Ended</i> , when it is configured to do so.
Prerequisite(s)	The Charging Station has an energy meter.

Before (Preparations)

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 <i>ParkingBayUnoccupied</i>	
<p><u>Note(s):</u></p> <ul style="list-style-type: none"> - This step will be executed after the <i><Configured transaction duration></i> is reached. - This causes the transaction to stop. 	

Tool validations

N/a

Post scenario validations:

- The **TransactionEventRequest** containing eventType *Ended* contains **meterValue** fields.
- The **meterValue** must contain an array of *<elements per data collection moment indicated by AlignedDataTxEndedInterval>*.
- **metervalue[0].timestamp** must be greater/equal then start time of transaction.
- *<The intervals between the timestamps of the meterValues must equal the configured value at AlignedDataTxEndedInterval.>*
- Last **metervalue[n].timestamp** must be smaller/equal then end time of transaction.
- **sampledValue[x].context** must be *Sample.Clock* for all **sampledValues**.
- **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">_
- None of the provided **sampledValues** shall have **location** = EV, except when **measurand** = SoC.

TC_J_04_CS: Clock-aligned Meter Values - Signed

Test case name	Clock-aligned Meter Values - Signed
Test case Id	TC_J_04_CS
Use case Id(s)	J01
Requirement(s)	J01.FR.21
System under test	Charging Station
Description	The Charging Station samples the electrical meter or other sensor/transducer hardware to provide information about its Meter Values. Depending on configuration settings, the Charging Station will send Meter Values.
Purpose	To verify if the Charging Station is able to send signed clock-aligned Meter Values when a transaction ends at the TransactionEventRequest with eventType is <i>Ended</i> , when it is configured to do so.
Prerequisite(s)	The Charging Station has an energy meter.

Before (Preparations)
Configuration State: AlignedDataTxEndedInterval is <i><Configured clock_aligned_tx_ended_meter_values_interval></i> AlignedDataSendDuringIdle is <i>false</i> (If implemented) AlignedDataSignReadings is <i>true</i>
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>ParkingBayUnoccupied</i>	
<u>Note(s):</u> - 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 <i>Ended</i> contains the MeterValue field. - The MeterValue should contain <i><An element per data collection moment indicated by AlignedDataTxEndedInterval. The Test System will not validate this.></i> - timestamp <i><The intervals between the timestamps of the received Meter Value messages should equal the configured value at AlignedDataTxEndedInterval.></i> - sampledValue[0].context should be <i>Sample.Clock</i> - sampledValue should contain <i><An element per configured measurand at the AlignedDataTxEndedMeasurands. The measurand field may be omitted when the measurand is "Energy.Active.Import.Register"></i> - 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 - None of the provided sampledValues shall have location = EV, except when measurand = SoC.

TC_J_06_CS: Clock-aligned Meter Values - No Meter Values during transaction

Test case name	Clock-aligned Meter Values - No Meter Values during transaction
Test case Id	TC_J_06_CS
Use case Id(s)	J01
Requirement(s)	N/a
System under test	Charging Station
Description	The Charging Station samples the electrical meter or other sensor/transducer hardware to provide information about its Meter Values. Depending on configuration settings, the Charging Station will send Meter Values.
Purpose	To verify if the Charging Station is able to only send clock-aligned Meter Values when there is no ongoing transaction, when it is configured to do so.
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station has an energy meter. - The configuration variable 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
<p>1. The Charging Station notifies the CSMS about its measured Meter Values.</p> <p><u>Note(s):</u></p> <ul style="list-style-type: none"> - 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 (evseld=0) 	<p>2. The Test System responds accordingly.</p>
<p>3. Execute Reusable State <i>EnergyTransferStarted</i></p>	
<p>4. The Charging Station notifies the CSMS about its measured Meter Values.</p> <p><u>Note(s):</u></p> <ul style="list-style-type: none"> - The Meter Value messages should not be send/received at the exact specified interval. 	<p>5. The Test System responds accordingly.</p>
<p>6. Execute Reusable State <i>ParkingBayUnoccupied</i></p> <p><u>Note(s):</u></p> <ul style="list-style-type: none"> - This step will be executed after the <Configured clock-aligned Meter Values interval + 5 seconds> is reached. 	

Main (Test scenario)	
<p>7. The Charging Station notifies the CSMS about its measured Meter Values.</p> <p><u>Note(s):</u></p> <ul style="list-style-type: none"> - The Meter Value messages do NOT need to be send/received at the exact specified interval. The configured measurands must be measured at the configured interval. - Multiple Meter Value messages may be sent per configured interval. One (or more in case the amount of measured data is too much for one message) for each EVSE and one (or more) for the main power meter (evseld=0) 	<p>8. The Test System responds accordingly.</p>

Tool validations
<p>* Step 1 & 7:</p> <p>Message: MeterValuesRequest</p> <ul style="list-style-type: none"> - sampledValue[0].context must be <i>Sample.Clock</i> - sampledValue must contain <An element per configured measurand at the <i>AlignedDataMeasurands</i>. The measurand field may be omitted when the measurand is "Energy.Active.Import.Register">
<p>Post scenario validations:</p> <p>Message: MeterValuesRequest</p> <ul style="list-style-type: none"> - timestamp <The intervals between the timestamps of the received Meter Value messages must equal the configured value at <i>AlignedDataInterval</i>. However it is allowed to send multiple Meter Value messages per configured interval. One (or more in case the amount of measured data is too much for one message) for each EVSE and one (or more) for the main power meter (evseld=0). But the timestamp of these messages must all be the same.> - The Charging Station did not send any message to report Meter Values to the Test System, 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. - None of the provided sampledValues shall have location = EV, except when measurand = SoC.

TC_J_07_CS: Sampled Meter Values - EventType Started - EVSE known

Test case name	Sampled Meter Values - EventType Started - EVSE known
Test case Id	TC_J_07_CS
Use case Id(s)	J02 & (E01,E02,E03,E09,E04,E05)
Requirement(s)	J02.FR.01,J02.FR.02,J02.FR.03,J02.FR.04,J02.FR.10,E01.FR.09,E02.FR.09,E03.FR.07,E04.FR.05,E05.FR.05
System under test	Charging Station
Description	The Charging Station samples the electrical meter or other sensor/transducer hardware to provide information about its Meter Values. Depending on configuration settings, the Charging Station will send Meter Values.
Purpose	To verify if the Charging Station is able to send sampled Meter Values when a transaction starts and the EVSE is known, at the TransactionEventRequest with eventType is <i>Started</i> , when it is configured to do so.
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station has an energy meter. - The Charging Station does NOT have the following configuration; TxStartPoint contains <i>ParkingBayOccupancy</i>

Before (Preparations)

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 <i>EVConnectedPreSession</i>	
2. Execute Reusable State <i>EnergyTransferStarted</i>	

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">
- None of the provided **sampledValues** shall have **location** = EV, except when **measurand** = SoC.

TC_J_08_CS: Sampled Meter Values - Context Transaction.Begin - EVSE not known

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)	<ul style="list-style-type: none"> - The Charging Station has an energy meter. - The Charging Station does NOT have the following configuration; TxStartPoint does NOT contain <i>ParkingBayOccupancy</i> OR <i>Authorized</i>.

Before (Preparations)
Configuration State: TxStartPoint contains <i>Authorized</i> Note: TxStartPoint contains <i>Authorized</i> AND/OR <i>ParkingBayOccupancy</i> (At least one of these values must be set).
Memory State: N/a
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>EnergyTransferStarted</i>	

Tool validations
N/a
Post scenario validations: <ul style="list-style-type: none"> - The first TransactionEventRequest containing a value for evse, sent during the execution of reusable state <i>EVConnectedPreSession</i> contains the MeterValue field with: <ul style="list-style-type: none"> - sampledValue[0].context must be <i>Transaction.Begin</i> - sampledValue must contain <An element per configured measurand at the <i>SampledDataTxStartedMeasurands</i>. The measurand field may be omitted when the measurand is "Energy.Active.Import.Register"> - None of the provided sampledValues shall have location = EV, except when measurand = SoC.

TC_J_09_CS: Sampled Meter Values - EventType Updated

Test case name	Sampled Meter Values - EventType Updated
Test case Id	TC_J_09_CS
Use case Id(s)	J02 & (E01,E02,E03,E09,E04,E05)
Requirement(s)	J02.FR.01, J02.FR.02, J02.FR.03, J02.FR.04, J02.FR.10, J02.FR.11, J02.FR.14, E02.FR.10, E02.FR.11, E03.FR.08, E03.FR.09, E04.FR.06, E04.FR.09, E11.FR.03, E11.FR.06, E12.FR.03, E12.FR.06
System under test	Charging Station
Description	The Charging Station samples the electrical meter or other sensor/transducer hardware to provide information about its Meter Values. Depending on configuration settings, the Charging Station will send Meter Values.
Purpose	To verify if the Charging Station is able to send sampled Meter Values during the transaction, at the TransactionEventRequest with eventType is <i>Updated</i> , when it is configured to do so.
Prerequisite(s)	The Charging Station has an energy meter.

Before (Preparations)
Configuration State: SampledDataTxUpdatedInterval is <Configured sampled Meter Values Updated interval>
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
<p>1. The Charging Station sends a TransactionEventRequest</p> <p><u>Note(s):</u></p> <ul style="list-style-type: none">- The <i>TransactionEventRequest</i> messages do NOT need to be send/received at the exact specified interval. The configured measurands must be measured at the configured interval.- Multiple <i>TransactionEventRequest</i> messages may be sent per configured interval, in case the amount of measured data is too much for one message.- The Test System will end the testcase after it has the _<Configured transaction duration> is reached._	<p>2. The Test System responds with a TransactionEventResponse</p>

Tool validations
* Step 1: Message: TransactionEventRequest - triggerReason must be <i>MeterValuePeriodic</i> - sampledValue[0].context must be <i>Sample.Periodic</i> - sampledValue must contain <An element per configured measurand at the SampledDataTxUpdatedMeasurands. The measurand field may be omitted when the measurand is "Energy.Active.Import.Register">
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.> - None of the provided sampledValues shall have location = EV, except when measurand = SoC.

TC_J_10_CS: Sampled Meter Values - EventType Ended

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, E07.FR.08,E08.FR.09,E09.FR.05,E10.FR.04,E12.FR.07
System under test	Charging Station
Description	The Charging Station samples the electrical meter or other sensor/transducer hardware to provide information about its Meter Values. Depending on configuration settings, the Charging Station will send Meter Values.
Purpose	To verify if the Charging Station is able to send sampled Meter Values when a transaction ends at the TransactionEventRequest with eventType is <i>Ended</i> , when it is configured to do so.
Prerequisite(s)	The Charging Station has an energy meter.

Before (Preparations)

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 <i>ParkingBayUnoccupied</i>	
<p><u>Note(s):</u></p> <ul style="list-style-type: none"> - This step will be executed after the <i><Configured transaction duration></i> is reached. - This causes the transaction to stop. 	

Tool validations

N/a
<p>Post scenario validations:</p> <ul style="list-style-type: none"> - The TransactionEventRequest containing eventType <i>Ended</i> contains the MeterValue field. - The MeterValue must contain <i><An element per data collection moment indicated by SampledDataTxEndedInterval. The Test System will not validate this.></i> - timestamp <i><The intervals between the timestamps of the received Meter Value messages must equal the configured value at SampledDataTxEndedInterval.></i> - sampledValue[0].context must be <i>Sample.Periodic</i> AND one must have <i>Transaction.End</i> - sampledValue must contain <i><An element per configured measurand at the SampledDataTxEndedMeasurands. The measurand field may be omitted when the measurand is "Energy.Active.Import.Register"></i> - None of the provided sampledValues shall have location = EV, except when measurand = SoC.

TC_J_11_CS: Sampled Meter Values - Signed

Test case name	Sampled Meter Values - Signed
Test case Id	TC_J_11_CS
Use case Id(s)	J02
Requirement(s)	J02.FR.21
System under test	Charging Station
Description	The Charging Station samples the electrical meter or other sensor/transducer hardware to provide information about its Meter Values. Depending on configuration settings, the Charging Station will send Meter Values.
Purpose	To verify if the Charging Station is able to send sampled Meter Values when a transaction ends at the TransactionEventRequest with eventType is <i>Ended</i> , when it is configured to do so.
Prerequisite(s)	The Charging Station has an energy meter.

Before (Preparations)

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 Test System 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
- None of the provided **sampledValues** shall have **location** = *EV*, except when **measurand** = *SoC*.

K Smart Charging

Determine Charging Profile Limit Multiplier

Not all chargers support setting limits in A or W. This can be configured with the configuration variable *<Configured chargingRateUnit>*. To calculate the limit to be used, the following rules must be followed:

If *<Configured chargingRateUnit>* is A, then *<limit multiplier>* is 1

If *<Configured chargingRateUnit>* is W and *<Configured numberPhases>* is 1, then *<limit multiplier>* is 230

If *<Configured chargingRateUnit>* is W and *<Configured numberPhases>* is 2, then *<limit multiplier>* is 460

If *<Configured chargingRateUnit>* is W and *<Configured numberPhases>* is 3, then *<limit multiplier>* is 690

Example 1

Given a test case is configured with:

```
<Configured chargingRateUnit> W
<Configured numberPhases> 2
```

When the test case specifies:

```
chargingProfile.chargingSchedule.chargingSchedulePeriod.limit 6 * <limit multiplier>
```

Then it should set

```
chargingProfile.chargingSchedule.chargingSchedulePeriod.limit 2760
```

Example 2

Given a test case is configured with:

```
<Configured chargingRateUnit> A
<Configured numberPhases> 3
```

When the test case specifies:

```
chargingProfile.chargingSchedule.chargingSchedulePeriod.limit 6 * <limit multiplier>
```

Then it should set

```
chargingProfile.chargingSchedule.chargingSchedulePeriod.limit 6
```

TC_K_01_CS: Set Charging Profile - TxDefaultProfile - Specific EVSE

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetChargingProfileResponse	1. The Test System sends a SetChargingProfileRequest with evseld <Configured evseld> 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 * <limit multiplier> Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier> chargingProfile.chargingSchedule.chargingSchedulePeriod.numberPhases <Configured numberPhases>
4. The Charging Station responds with a GetChargingProfilesResponse	3. The Test System sends a GetChargingProfilesRequest with chargingProfile.chargingProfileId <Configured chargingProfileId> requestId <Generated requestId>
5. The Charging Station sends a ReportChargingProfilesRequest	6. The Test System 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 <Generated requestId> - evseld <Configured EVSEId> * - chargingProfile <Configured ChargingProfile>
Post scenario validations: - The same profile is reported as send in step 1

TC_K_02_CS: Set Charging Profile - TxProfile without ongoing transaction on the specified EVSE

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetChargingProfileResponse	1. The Test System sends a SetChargingProfileRequest with evseld <Configured evseld> AND chargingProfile.id <Configured chargingProfileId> AND chargingProfile.chargingProfilePurpose TxProfile AND chargingProfile.transactionId UNKNOWN-TRANSACTION-ID

Tool validations
* Step 2: Message SetChargingProfileResponse - status <i>Rejected</i>
Post scenario validations: - N/a

TC_K_03_CS: Set Charging Profile - ChargingStationMaxProfile

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

Reusable State:

N/a

Main (Test scenario)

Charging Station	CSMS
2. The Charging Station responds with a SetChargingProfileResponse	1. The Test System sends a SetChargingProfileRequest with chargingProfile.id <Configured chargingProfileId> AND chargingProfile.chargingProfilePurpose <i>ChargingStationMaxProfile</i> AND chargingProfile.chargingProfileKind <i>Absolute</i> AND chargingProfile.chargingSchedule.duration <Configured duration> AND chargingProfile.chargingSchedule.chargingRateUnit <Configured chargingRateUnit> AND chargingProfile.chargingSchedule.chargingSchedulePeriod.startPeriod 0 AND chargingProfile.chargingSchedule.chargingSchedulePeriod.limit 6 * <limit multiplier> AND <i>Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></i> chargingProfile.chargingSchedule.chargingSchedulePeriod.numberPhases <Configured numberPhases> AND EVSEId 0
4. The Charging Station responds with a GetChargingProfilesResponse	3. The Test System sends a GetChargingProfilesRequest with chargingProfile.chargingProfileId <Configured chargingProfileId> requestId <Generated requestId>
5. The Charging Station sends a ReportChargingProfilesRequest	6. The Test System responds with a ReportChargingProfilesResponse
Note(s): - If tbc is True at Step 5 then step 5 and 6 will be repeated	

Tool validations
<p>* Step 2:</p> <p>Message SetChargingProfileResponse</p> <ul style="list-style-type: none">- status <i>Accepted</i> <p>* Step 4:</p> <p>Message GetChargingProfilesResponse</p> <ul style="list-style-type: none">- status <i>Accepted</i> <p>* Step 5:</p> <p>Message ReportChargingProfilesRequest</p> <ul style="list-style-type: none">- requestId <i><Generated requestId></i>- EvseId <i>0</i>- chargingProfile <i><Generated chargingProfile></i>
<p>Post scenario validations:</p> <ul style="list-style-type: none">- The same profile is reported as send in step 1

TC_K_04_CS: Replace charging profile - With chargingProfileId

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetChargingProfileResponse	1. The Test System sends a SetChargingProfileRequest with chargingProfile.id <Configured chargingProfileId> chargingProfile.chargingProfilePurpose TxDefaultProfile chargingProfile.chargingSchedule.chargingSchedulePeriod.limit 10 * <limit multiplier> Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier>
4. The Charging Station responds with a GetChargingProfilesResponse	3. The Test System sends a GetChargingProfilesRequest with chargingProfile.chargingProfileId <Configured chargingProfileId>
5. The Charging Station sends a ReportChargingProfilesRequest	6. The Test System responds with a ReportChargingProfilesResponse
<u>Note(s):</u> - If tb 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>

Tool validations
Post scenario validations: - N/a

TC_K_05_CS: Clear Charging Profile - With chargingProfileId

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

Reusable State:

N/a

Main (Test scenario)

Charging Station	CSMS
2. The Charging Station responds with a ClearChargingProfileResponse	1. The Test System sends a ClearChargingProfileRequest with chargingProfileId <Configured chargingProfileId>
4. The Charging Station responds with a GetChargingProfilesResponse	3. The Test System 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

TC_K_06_CS: Clear Charging Profile - With stackLevel/purpose combination for one profile

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
Reusable State: EnergyTransferStarted

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a ClearChargingProfileResponse	1. The Test System sends a ClearChargingProfileRequest with chargingProfileCriteria.chargingProfilePurpose <Configured chargingProfilePurpose> AND chargingProfileCriteria.stackLevel <Configured stackLevel>
4. The Charging Station responds with a GetChargingProfilesResponse	3. The Test System sends a GetChargingProfilesRequest with chargingProfile.chargingProfilePurpose <Configured chargingProfilePurpose> AND chargingProfile.stackLevel <Configured stackLevel>

Tool validations
* Step 2: Message ClearChargingProfileResponse - status <i>Accepted</i>
* Step 4: Message GetChargingProfilesResponse - status <i>NoProfiles</i>
Post scenario validations: - N/a

TC_K_07_CS: Clear Charging Profile - With unknown stackLevel/purpose combination

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a ClearChargingProfileResponse	1. The Test System sends a ClearChargingProfileRequest with chargingProfileCriteria.chargingProfilePurpose <i>ChargingStationMaxProfile</i> AND chargingProfileCriteria.stackLevel 0

Tool validations
* Step 2: Message ClearChargingProfileResponse - status <i>Unknown</i>
Post scenario validations: - N/a

TC_K_08_CS: Clear Charging Profile - Without previous charging profile

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a ClearChargingProfileResponse	1. The Test System sends a ClearChargingProfileRequest with chargingProfileId <Generated <i>chargingProfileId</i> >

Tool validations
* Step 2: Message ClearChargingProfileResponse - status <i>Unknown</i>
Post scenario validations: - N/a

TC_K_09_CS: Clear Charging Profile - Clearing a TxDefaultProfile - With ongoing transaction

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 evseld <Configured evseld> startSchedule <current dateTime> numberPhases <Configured numberPhases> ChargingSchedule: duration 400 + <Configured max time deviation> chargingRateUnit <Configured chargingRateUnit> startPeriod 0, limit 6 * <limit multiplier> Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier>
Reusable State: State is EnergyTransferStarted

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a GetCompositeScheduleResponse	1. The Test System sends a GetCompositeScheduleRequest with evseld is <Configured evseld>
4. The Charging Station responds with a ClearChargingProfileResponse	3. The Test System sends a ClearChargingProfileRequest with chargingProfileCriteria.chargingProfilePurpose TxDefaultProfile
5. The Charging Station responds with a GetCompositeScheduleResponse	6. The Test System sends a GetCompositeScheduleRequest with evseld <Configured evseld> duration is 300 chargingRateUnit <Configured chargingRateUnit>

Tool validations
<p>* Step 2:</p> <p>(Message: GetCompositeScheduleResponse)</p> <p>status <i>Accepted</i></p> <p>evseld <i><Configured evseld></i></p> <p>ChargingSchedule:</p> <p>duration <i>300</i></p> <p>chargingRateUnit <i><Configured chargingRateUnit></i></p> <p>scheduleStart <i><The time the GetCompositeScheduleRequest was transmitted +/- <Configured max time deviation>></i></p> <p>startPeriod <i>0, limit 6 * <limit multiplier></i></p> <p><i>Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></i></p> <p>* Step 4:</p> <p>(Message: ClearChargingProfileResponse)</p> <p>status <i>is Accepted</i></p> <p>* Step 5:</p> <p>(Message: GetCompositeScheduleResponse)</p> <p>status <i>Accepted</i></p> <p>evseld <i><Configured evseld></i></p> <p>ChargingSchedule:</p> <p>duration <i>300</i></p> <p>chargingRateUnit <i><Configured chargingRateUnit></i></p> <p>scheduleStart <i><The time the GetCompositeScheduleRequest was transmitted +/- <Configured max time deviation>></i></p> <p>startPeriod <i>0, limit <Local limit of Charging Station (> 6 * <limit multiplier>)></i></p> <p><i>Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></i></p>
<p>Post scenario validations:</p> <p>N/a</p>

TC_K_10_CS: Set Charging Profile - TxDefaultProfile - All EVSE

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

Reusable State:

N/a

Main (Test scenario)

Charging Station	CSMS
2. The Charging Station responds with a SetChargingProfileResponse	<p>1. The Test System sends a SetChargingProfileRequest with evseld 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 * <limit multiplier> Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier> chargingProfile.chargingSchedule.chargingSchedulePeriod.numberPhases <Configured numberPhases></p>
4. The Charging Station responds with a GetChargingProfilesResponse	3. The Test System sends a GetChargingProfilesRequest with chargingProfile.chargingProfileId <Configured chargingProfileId> requestId <Generated requestId>
5. The Charging Station sends a ReportChargingProfilesRequest	6. The Test System responds with a ReportChargingProfilesResponse
Note(s): - If tbc is True at Step 5 then step 5 and 6 will be repeated	

Tool validations
<p>* Step 2:</p> <p>Message SetChargingProfileResponse</p> <ul style="list-style-type: none">- status <i>Accepted</i> <p>* Step 4:</p> <p>Message GetChargingProfilesResponse</p> <ul style="list-style-type: none">- status <i>Accepted</i> <p>* Step 5:</p> <p>Message ReportChargingProfilesRequest</p> <ul style="list-style-type: none">- requestId <i><Generated requestId></i>- EVSEId <i>0</i>- tbc <i>false</i>- chargingProfile <i><Configured chargingProfile></i>
<p>Post scenario validations:</p> <ul style="list-style-type: none">- The same profile is reported as send in step 1

TC_K_11_CS: Set Charging Profile - Unable to set TxProfile on all EVSE at once

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetChargingProfileResponse	1. The Test System sends a SetChargingProfileRequest with evseld 0 AND chargingProfile.id <Configured chargingProfileId> AND chargingProfile.chargingProfilePurpose TxProfile

Tool validations
* Step 2: Message SetChargingProfileResponse - status Rejected
Post scenario validations: - N/a

TC_K_12_CS: Set Charging Profile - ChargerRateUnit Rejected

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

Reusable State:
N/a

Main (Test scenario)

Charging Station	CSMS
2. The Charging Station responds with a SetChargingProfileResponse	1. The Test System 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

TC_K_13_CS: Set Charging Profile - Persistent over reboot

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

Reusable State:

N/a

Main (Test scenario)

Charging Station	CSMS
2. The Charging Station responds with a SetChargingProfileResponse	<p>1. The Test System sends a SetChargingProfileRequest with evseld <Configured evseld> 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 * <limit multiplier> Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier> chargingProfile.chargingSchedule.chargingSchedulePeriod.numberPhases <Configured numberPhases></p>
3. Execute Reusable State <i>Booted</i>	
5. The Charging Station responds with a GetChargingProfilesResponse	4. The Test System sends a GetChargingProfilesRequest with chargingProfile.chargingProfileId <Configured chargingProfileId>
6. The Charging Station sends a ReportChargingProfilesRequest	7. The Test System responds with a ReportChargingProfilesResponse
Note(s): - If tbc is True at Step 6 then step 6 and 7 will be repeated	

Tool validations
<p>* Step 2:</p> <p>Message SetChargingProfileResponse</p> <ul style="list-style-type: none">- status <i>Accepted</i> <p>* Step 5:</p> <p>Message GetChargingProfilesResponse</p> <ul style="list-style-type: none">- status <i>Accepted</i> <p>* Step 6:</p> <p>Message ReportChargingProfilesRequest</p> <ul style="list-style-type: none">- requestId <i>Same Id as in the GetChargingProfilesRequest in step 4</i>- EVSEId <i><Configured EVSEId></i>- chargingProfile <i><Configured chargingProfile></i>
<p>Post scenario validations:</p> <ul style="list-style-type: none">- The same profile is reported as send in step 1

TC_K_14_CS: Set Charging Profile - Unexisting EVSEid

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetChargingProfileResponse	1. The Test System sends a SetChargingProfileRequest with evseld <EVSECount + 1>

Tool validations
* Step 2: Message SetChargingProfileResponse - status <i>Rejected</i>
Post scenario validations: - N/a

TC_K_15_CS: Set Charging Profile - Not Supported

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with RPC Framework: CALLERROR: NotSupported.	1. The Test System sends a SetChargingProfileRequest with chargingProfile.id <Configured chargingProfileId>

Tool validations
- N/a
Post scenario validations: - N/a

TC_K_16_CS: Set Charging Profile - Unknown transactionId

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
Reusable State: State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetChargingProfileResponse	1. The Test System sends a SetChargingProfileRequest with evseld <Configured evseld> 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

TC_K_19_CS: Set Charging Profile - ChargingProfileKind is Recurring

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetChargingProfileResponse	1. The Test System sends a SetChargingProfileRequest with chargingProfile.chargingProfileKind Recurring chargingProfile.recurrencyKind <Configured RecurrencyKind>

Tool validations
* Step 2: Message SetChargingProfileResponse - status Accepted
Post scenario validations: - N/a

TC_K_21_CS: Set Charging Profile - ValidFrom

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	<p>1. The Test System sends a SetChargingProfileRequest with chargingProfile.chargingProfilePurpose is <i>TxDefaultProfile</i> chargingProfile.chargingProfileKind is <i>Relative</i> evseld <configured evseld> 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 chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].limit 6 * <limit multiplier> Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></p>
4. The Charging Station responds with a GetCompositeScheduleResponse	<p>3. The Test System sends a GetCompositeScheduleRequest with evseld <configured evseld> duration is 400 chargingRateUnit <Configured chargingRateUnit></p>

Tool validations
<p>* Step 2: (Message: SetChargingProfileResponse) status is <i>Accepted</i></p> <p>* Step 4: (Message: GetCompositeScheduleResponse) status <i>Accepted</i> schedule.evseId <i><configured evseId></i> schedule.chargingRateUnit <i><Configured chargingRateUnit></i> schedule.duration <i>400</i> schedule.scheduleStart <i><The time the GetCompositeScheduleRequest was transmitted +/- <Configured max time deviation>></i> schedule.chargingSchedulePeriod[0].startPeriod <i>0</i> schedule.chargingSchedulePeriod[0].limit <i><Local limit of Charging Station (> 6)></i> schedule.chargingSchedulePeriod[1].startPeriod <i>(300 - x)</i> schedule.chargingSchedulePeriod[1].limit <i>6.0 * <limit multiplier></i></p> <p>Note: Check Determine Charging Profile Limit Multiplier for <i><limit multiplier></i> Note: The period of time between the <i>scheduleStart</i> from the <i>SetChargingProfileRequest</i> and the <i>scheduleStart</i> from the <i>GetCompositeScheduleResponse</i> is called <i>x</i>.</p>
<p>Post scenario validations: N/a</p>

TC_K_22_CS: Set Charging Profile - ValidTo

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	<p>1. The Test System sends a SetChargingProfileRequest with chargingProfile.chargingProfilePurpose is <i>TxDefaultProfile</i> chargingProfile.chargingProfileKind is <i>Absolute</i> evseld <Configured evseld> chargingProfile.validFrom <current dateTime - <Configured max time deviation> seconds> chargingProfile.validTo <current dateTime + 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 chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].limit 6 * <limit multiplier> <i>Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></i></p>
4. The Charging Station responds with a GetCompositeScheduleResponse	<p>3. The Test System sends a GetCompositeScheduleRequest with evseld 1 duration is 400 chargingRateUnit <Configured chargingRateUnit></p>

Tool validations
<p>* Step 2: (Message: SetChargingProfileResponse) status is <i>Accepted</i></p> <p>* Step 4: (Message: GetCompositeScheduleResponse) status <i>Accepted</i> evseld <Configured evseld> chargingRateUnit <Configured chargingRateUnit> ChargingSchedule: duration 400 chargingRateUnit <Configured chargingRateUnit> scheduleStart <The time the GetCompositeScheduleRequest was transmitted +/- <Configured max time deviation>> <i>Note: The period of time between the <code>scheduleStart</code> from the SetChargingProfileRequest and the <code>scheduleStart</code> from the GetCompositeScheduleResponse is called x.</i> startPeriod 0, limit 6 * <limit multiplier> startPeriod (300 - x), limit <Local limit of Charging Station (> 6 * <limit multiplier>)> <i>Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></i></p>
<p>Post scenario validations: N/a</p>

TC_K_23_CS: Set Charging Profile - StartSchedule

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	<p>1. The Test System sends a SetChargingProfileRequest with chargingProfile.chargingProfilePurpose is <i>TxDefaultProfile</i> chargingProfile.chargingProfileKind is <i>Absolute</i> evseld <configured evseld> chargingProfile.chargingSchedule[0].startSchedule <current dateTime> + 60 seconds> chargingProfile.chargingSchedule[0].chargingSchedulePeriod[].numberPhases <Configured numberPhases> chargingProfile.chargingSchedule[0].chargingSchedulePeriod[].startPeriod 0 chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].limit 6 * <limit multiplier> Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></p>
4. The Charging Station responds with a GetCompositeScheduleResponse	<p>3. The Test System sends a GetCompositeScheduleRequest with evseld <Configured evseld> duration is 300 chargingRateUnit <Configured chargingRateUnit></p>

Tool validations
<p>* Step 2: (Message: SetChargingProfileResponse) status is <i>Accepted</i></p> <p>* Step 4: (Message: GetCompositeScheduleResponse) status <i>Accepted</i> evseld <i><Configured evseld></i> ChargingSchedule: duration <i>300</i> chargingRateUnit <i><Configured chargingRateUnit></i> scheduleStart <i><The time the GetCompositeScheduleRequest was transmitted +/- <Configured max time deviation>></i> <i>Note: The period of time between the <code>scheduleStart</code> from the SetChargingProfileRequest and the <code>scheduleStart</code> from the GetCompositeScheduleResponse is called x.</i> startPeriod <i>0</i>, limit <i><Local limit of Charging Station (> 6 * <limit multiplier>)></i> startPeriod <i>(60 - x)</i>, limit <i>6 * <limit multiplier></i> <i>Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></i></p>
<p>Post scenario validations: N/a</p>

TC_K_24_CS: Clear Charging Profile - With stackLevel/purpose combination for multiple profiles

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 evseld 1. A chargingprofile with <Configured chargingProfilePurpose> AND <Configured stackLevel> is configured for evseld 2.
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a ClearChargingProfileResponse	1. The Test System sends a ClearChargingProfileRequest with chargingProfileCriteria.chargingProfilePurpose <Configured chargingProfilePurpose> AND chargingProfileCriteria.stackLevel <Configured stackLevel>
4. The Charging Station responds with a GetChargingProfilesResponse	3. The Test System sends a GetChargingProfilesRequest with chargingProfile.chargingProfilePurpose <Configured chargingProfilePurpose> AND chargingProfile.stackLevel <Configured stackLevel>

Tool validations
* Step 2: Message ClearChargingProfileResponse - status <i>Accepted</i> * Step 4: Message GetChargingProfilesResponse - status <i>NoProfiles</i>
Post scenario validations: - N/a

TC_K_28_CS: Set Charging Profile - TxDefaultProfile with transaction ongoing

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	<p>1. The Test System sends a SetChargingProfileRequest with chargingProfile.chargingProfilePurpose is <i>TxDefaultProfile</i> chargingProfile.chargingProfileKind is <i>Absolute</i> chargingProfile.chargingSchedule[0].duration is $<400 + \text{<Configured max time deviation> seconds}>$ evseld <i><Configured evseld></i> chargingProfile.chargingSchedule[0].startSchedule <i><current dateTime - <Configured max time deviation> seconds></i> chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].numberPhases <i><Configured numberPhases></i> chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].startPeriod 0 chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].limit $6 * \text{<limit multiplier>}$ <i>Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></i></p>
4. The Charging Station responds with a GetCompositeScheduleResponse	<p>3. The Test System sends a GetCompositeScheduleRequest with evseld <i><Configured evseld></i> duration is 300 chargingRateUnit <i><Configured chargingRateUnit></i></p>

Tool validations
<p>* Step 2: (Message: SetChargingProfileResponse) status is <i>Accepted</i></p> <p>* Step 4: (Message: GetCompositeScheduleResponse) status <i>Accepted</i> evseld <i><Configured evseld></i> ChargingSchedule: duration <i>300</i> chargingRateUnit <i><Configured chargingRateUnit></i> scheduleStart <i><The time the GetCompositeScheduleRequest was transmitted +/- <Configured max time deviation>></i> startPeriod <i>0</i>, limit <i>6 * <limit multiplier></i> <i>Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></i></p>
<p>Post scenario validations: N/a</p>

TC_K_29_CS: Get Charging Profile - Evseld 0

Test case name	Get Charging Profile - Evseld 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 <i>ChargingStationMaxProfile</i> configured on the charging station. Charging station has a second charge profile with <Generated Id2> AND chargingProfilePurpose <i>TxDefaultProfile</i> configured on <Configured evseld>.
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a GetChargingProfilesResponse	1. The Test System sends a GetChargingProfilesRequest with evseld 0
3. The Charging Station sends a ReportChargingProfilesRequest	4. The Test System 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 <i>Accepted</i> * Step 3: Message ReportChargingProfilesRequest - requestId <Generated requestId> - chargingProfile <Generated ChargingProfile1> with chargingProfilePurpose <i>ChargingStationMaxProfile</i>
Post scenario validations: - All report message have been received

TC_K_30_CS: Get Charging Profile - Evseld > 0

Test case name	Get Charging Profile - Evseld > 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 <i>ChargingStationMaxProfile</i> configured on the charging station. Charging station has a second charge profile with <Generated Id2> AND <i>TxDefaultProfile</i> configured on <Configured evseld>.
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a GetChargingProfilesResponse	1. The Test System sends a GetChargingProfilesRequest with evseld <Configured evseld>
3. The Charging Station sends a ReportChargingProfilesRequest	4. The Test System 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 <i>Accepted</i> * Step 3: Message ReportChargingProfilesRequest - requestId <Generated requestId> - chargingProfile <Generated ChargingProfile>
Post scenario validations: - All report message have been received

TC_K_31_CS: Get Charging Profile - No Evseld

Test case name	Get Charging Profile - No Evseld
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 evseld>.
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a GetChargingProfilesResponse	1. The Test System sends a GetChargingProfilesRequest with: requestId <i>Generated requestId</i>
3. The Charging Station sends a ReportChargingProfilesRequest	4. The Test System 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 <i>Accepted</i> * Step 3: Message ReportChargingProfilesRequest - requestId <Generated requestId> - chargingProfiles <Configured ChargingProfiles>
Post scenario validations: - All report message have been received

TC_K_32_CS: Get Charging Profile - chargingProfileId

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>.
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a GetChargingProfilesResponse	1. The Test System sends a GetChargingProfilesRequest with chargingProfileId <Generated Id1>
3. The Charging Station sends a ReportChargingProfilesRequest	4. The Test System 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

TC_K_33_CS: Get Charging Profile - Evseld > 0 + stackLevel

Test case name	Get Charging Profile - Evseld > 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 evseld>.
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a GetChargingProfilesResponse	1. The Test System sends a GetChargingProfilesRequest with evseld <Configured evseld> AND chargingProfile.stackLevel <Configured stackLevel>
3. The Charging Station sends a ReportChargingProfilesRequest	4. The Test System responds with a ReportChargingProfilesResponse
<u>Note(s):</u> - 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

TC_K_34_CS: Get Charging Profile - Evseld > 0 + chargingLimitSource

Test case name	Get Charging Profile - Evseld > 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>
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a GetChargingProfilesResponse	1. The Test System sends a GetChargingProfilesRequest with evseld <Configured evseld> AND chargingProfile.chargingLimitSource <Configured chargingLimitSource>
3. The Charging Station sends a ReportChargingProfilesRequest	4. The Test System 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 Test System sends a GetChargingProfilesRequest with evseld <Configured evseld> 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

Tool validations
Post scenario validations: - All report message have been received

TC_K_35_CS: Get Charging Profile - Evseld > 0 + chargingProfilePurpose

Test case name	Get Charging Profile - Evseld > 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 evseld>.
Reusable State: State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a GetChargingProfilesResponse	1. The Test System sends a GetChargingProfilesRequest with evseld <Configured evseld> AND chargingProfile.chargingProfilePurpose <Configured chargingProfilePurpose>
3. The Charging Station sends a ReportChargingProfilesRequest	4. The Test System 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 <i>Accepted</i> * Step 3: Message ReportChargingProfilesRequest - requestId <i>Generated Id1</i> - ChargingProfile <Configured ChargingProfile>
Post scenario validations: - All report message have been received

TC_K_36_CS: Get Charging Profile - Evseld > 0 + chargingProfilePurpose + stackLevel

Test case name	Get Charging Profile - Evseld > 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 evseld>.
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a GetChargingProfilesResponse	1. The Test System sends a GetChargingProfilesRequest with evseld <Configured evseld> AND chargingProfile.chargingProfilePurpose <TxDefaultProfile> AND chargingProfile.stackLevel <Configured stackLevel>
3. The Charging Station sends a ReportChargingProfilesRequest	4. The Test System 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

TC_K_60_CS: Set Charging Profile - TxProfile with ongoing transaction on the specified EVSE

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 <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetChargingProfileResponse	<p>1. The Test System sends a SetChargingProfileRequest with chargingProfile.chargingProfilePurpose is <i>TxProfile</i> chargingProfile.transactionId is <i><transactionId returned by Charging Station in before></i> chargingProfile.chargingProfileKind is <i>Relative</i> evseld <i><Configured evseld></i> chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].numberPhases <i><Configured numberPhases></i> chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].startPeriod <i>0</i> chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].limit <i>6 * <limit multiplier></i> Note: Check <i>Determine Charging Profile Limit Multiplier</i> for <i><limit multiplier></i></p>
4. The Charging Station responds with a GetChargingProfilesResponse	<p>3. The Test System sends a GetChargingProfilesRequest with chargingProfile.chargingProfileId <i><Used chargingProfileId at step 1></i></p>
5. The Charging Station sends a ReportChargingProfilesRequest	<p>6. The Test System responds with a ReportChargingProfilesResponse</p>

Tool validations
<p>* Step 2: (Message: SetChargingProfileResponse) status is <i>Accepted</i></p> <p>* Step 4: (Message: GetChargingProfilesResponse) status is <i>Accepted</i></p> <p>* Step 5: (Message: ReportChargingProfilesRequest) chargingProfile <i><The Charging Profile set at step 1></i></p>

Tool validations
Post scenario validations: N/a

TC_K_37_CS: Remote start transaction with charging profile - Success

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 <i>EVConnectedPreSession</i>

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a RequestStartTransactionResponse	<p>1. The Test System sends a RequestStartTransactionRequest with idToken.idToken <Configured valid_idtoken_idtoken> idToken.type <Configured valid_idtoken_type> evseld <Configured evseld> chargingProfile.chargingProfilePurpose is <i>TxProfile</i> chargingProfile.transactionId is omitted. chargingProfile.chargingProfileKind is <i>Relative</i> chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].numberPhases <Configured numberPhases> chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].startPeriod 0 chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].limit 6 * <limit multiplier> Note: Check <i>Determine Charging Profile Limit Multiplier</i> for <limit multiplier></p>
<p>3. The Charging Station sends an AuthorizeRequest</p> <p><u>Note(s):</u> - This step needs to be executed when AuthCtrlr.AuthorizeRemoteStart is true, unless (AuthEnabled is implemented with mutability <i>ReadOnly</i> AND the value is set to false) OR the idToken is cached. In case the idToken is used for a reservation, sending the AuthorizeRequest message is optional.</p>	<p>4. The Test System responds with an AuthorizeResponse with idTokenInfo.status <i>Accepted</i></p>
<p>5. The Charging Station sends a TransactionEventRequest</p> <p><u>Note(s):</u> - This step needs to be executed when TxStartPoint contains <i>Authorized</i> OR the transaction already started. So in the case TxStartPoint contains <i>ParkingBayOccupancy</i> or (<i>EVConnected</i>, in the case this testcase was initiated from state <i>EVConnectedPreSession</i>.)</p>	<p>6. The Test System responds with a TransactionEventResponse</p> <p><u>Note(s):</u> - The first TransactionEventRequest sent after authorization contains the idToken field. The TransactionEventResponse of this request message contains idTokenInfo with status <i>Accepted</i></p>

Main (Test scenario)	
7. Execute Reusable State <i>EnergyTransferStarted</i>	
9. The Charging Station responds with a GetChargingProfilesResponse	8. The Test System sends a GetChargingProfilesRequest with chargingProfile.chargingProfileId <Used chargingProfileId at step 1>
10. The Charging Station sends a ReportChargingProfilesRequest	11. The Test System responds with a ReportChargingProfilesResponse
Tool validations	
<p>* Step 2: Message: RequestStartTransactionResponse - status must be <i>Accepted</i> If the transaction has already been started, so if TxStartPoint contains <i>ParkingBayOccupancy</i> OR (<Configured TxStartPoint> contains <i>EVConnected</i> AND State pre reusable state execution was <i>EVConnectedPreSession</i>) then - transactionId must be <Provided transactionId in first TransactionEventRequest></p> <p>* Step 3: Message: AuthorizeRequest - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type></p> <p>* Step 5: Message: TransactionEventRequest - triggerReason must be <i>RemoteStart</i> - transactionInfo.remoteStartId must be present.</p> <p>* Step 9: (Message: GetChargingProfilesResponse) status is <i>Accepted</i></p> <p>* Step 10: (Message: ReportChargingProfilesRequest) chargingProfile <The Charging Profile set at step 1></p>	
Post scenario validations: N/a	

TC_K_38_CS: Remote start transaction with charging profile - Ignore chargingProfile

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 <i>EVConnectedPreSession</i>

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a RequestStartTransactionResponse	<p>1. The Test System sends a RequestStartTransactionRequest with idToken.idToken <Configured valid_idtoken_idtoken> idToken.type <Configured valid_idtoken_type> evseld <Configured evseld> 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 chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].limit 6 * <limit multiplier> Note: Check <i>Determine Charging Profile Limit Multiplier</i> for <limit multiplier></p>
<p>3. The Charging Station sends an AuthorizeRequest</p> <p><u>Note(s):</u> - This step needs to be executed when AuthCtrlr.AuthorizeRemoteStart is true, unless (AuthEnabled is implemented with mutability ReadOnly AND the value is set to false) OR the idToken is cached. In case the idToken is used for a reservation, sending the AuthorizeRequest message is optional.</p>	<p>4. The Test System responds with an AuthorizeResponse with idTokenInfo.status Accepted</p>

Main (Test scenario)	
<p>5. The Charging Station sends a TransactionEventRequest</p> <p><u>Note(s):</u> - This step needs to be executed when TxStartPoint contains Authorized OR the transaction already started. So in the case TxStartPoint contains <i>ParkingBayOccupancy</i> or (<i>EVConnected</i>, in the case this testcase was initiated from state <i>EVConnectedPreSession</i>.)</p>	<p>6. The Test System responds with a TransactionEventResponse</p> <p><u>Note(s):</u> - The first <i>TransactionEventRequest</i> sent after authorization contains the <i>idToken</i> field. The <i>TransactionEventResponse</i> of this request message contains idTokenInfo with status Accepted</p>
<p>7. Execute Reusable State <i>EnergyTransferStarted</i></p>	

Tool validations
<p>* Step 2: Message: RequestStartTransactionResponse - status must be <i>Accepted</i> If the transaction has already been started, so if <i>TxStartPoint</i> contains <i>ParkingBayOccupancy</i> OR (<i>TxStartPoint</i> contains <i>EVConnected</i> AND State pre reusable state execution was <i>EVConnectedPreSession</i>) then - transactionId must be <i><Provided transactionId in first TransactionEventRequest></i></p> <p>* Step 3: Message: AuthorizeRequest - idToken.idToken <i><Configured valid_idtoken_idtoken></i> - idToken.type <i><Configured valid_idtoken_type></i></p> <p>* Step 5: Message: TransactionEventRequest - triggerReason must be <i>RemoteStart</i> - transactionInfo.remoteStartId must be present.</p>
<p>Post scenario validations: N/a</p>

TC_K_39_CS: Get Composite Schedule - No ChargingProfile installed on Charging Station

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 Test System sends a GetCompositeScheduleRequest with evseld 0 duration is 300 chargingRateUnit <Configured chargingRateUnit>

Tool validations
<p>* Step 2: (Message: GetCompositeScheduleResponse) status Accepted evseld 0 scheduleStart <The time the GetCompositeScheduleRequest was transmitted +/- <Configured max time deviation>> duration is 300 chargingRateUnit <Configured chargingRateUnit> startPeriod 0</p>
Post scenario validations: N/a

TC_K_40_CS: Get Composite Schedule - Stacking ChargingProfiles

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. 2 ChargingProfiles with same startSchedule and different stackLevels are submitted.
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)	<ul style="list-style-type: none"> - 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:

set <startScheduleTime> = <current dateTime> - <Configured max time deviation>

[SetChargingProfile](#) with

ChargingProfile 1:

chargingProfilePurpose is *TxDefaultProfile*

chargingProfileKind should be *Absolute*

stackLevel should be 0

evseld <Configured evseld>

startSchedule <startScheduleTime>

numberPhases <Configured numberPhases>

ChargingSchedule:

duration 400

chargingRateUnit <Configured chargingRateUnit>

startPeriod 0, **limit** 6 * <limit multiplier>

startPeriod 100, **limit** 8 * <limit multiplier>

startPeriod 200, **limit** 10 * <limit multiplier>

Note: Check [Determine Charging Profile Limit Multiplier](#) for <limit multiplier>

ChargingProfile 2:

chargingProfilePurpose is *TxDefaultProfile*

chargingProfileKind should be *Absolute*

stackLevel should be 1

evseld <Configured evseld>

startSchedule <startScheduleTime>

numberPhases <Configured numberPhases>

ChargingSchedule:

duration 150

chargingRateUnit <Configured chargingRateUnit>

startPeriod 0, **limit** 7 * <limit multiplier>

Note: Check [Determine Charging Profile Limit Multiplier](#) for <limit multiplier>

startPeriod 100, **limit** 9 * <limit multiplier>

Note: Check [Determine Charging Profile Limit Multiplier](#) for <limit multiplier>

Reusable State(s):

N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a GetCompositeScheduleResponse	1. The Test System sends a GetCompositeScheduleRequest with evseld <Configured evseld> duration is 350 chargingRateUnit <Configured chargingRateUnit>

Tool validations
<p>* Step 2: (Message: GetCompositeScheduleResponse) status <i>Accepted</i> evseld <Configured evseld> ChargingSchedule: duration 350 chargingRateUnit <Configured chargingRateUnit> scheduleStart <The time the GetCompositeScheduleRequest was transmitted> plus/minus <Configured max time deviation> <i>Note: The period of time between the scheduleStart from the SetChargingProfileRequest and the scheduleStart from the GetCompositeScheduleResponse is called x.</i> startPeriod 0, limit $7 * \text{<limit multiplier>}$ (stackLevel 1) startPeriod (100 - x), limit $9 * \text{<limit multiplier>}$ (stackLevel 1) startPeriod (150 - x), limit $8 * \text{<limit multiplier>}$ (stackLevel 0) startPeriod (200 - x), limit $10 * \text{<limit multiplier>}$ (stackLevel 0) <i>Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></i></p>
<p>Post scenario validations: N/a</p>

TC_K_41_CS: Get Composite Schedule - Combining chargingProfilePurposes

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. 3 ChargingProfiles are submitted with the same startSchedule.
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: set <startScheduleTime> = <current dateTime> - <Configured max time deviation> seconds Note: Set MaxProfile for the next 24 hours: SetChargingProfile with ChargingProfile 1: chargingProfilePurpose is <i>ChargingStationMaxProfile</i> chargingProfileKind should be <i>Absolute</i> stackLevel should be 0 evseld 0 startSchedule <startScheduleTime> numberPhases <Configured numberPhases> ChargingSchedule: duration 86400 chargingRateUnit <Configured chargingRateUnit> startPeriod 0, limit 10 * <limit multiplier> Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier>	
Note: Set a default profile for 300 seconds ChargingProfile 2: chargingProfilePurpose is <i>TxDefaultProfile</i> chargingProfileKind should be <i>Absolute</i> stackLevel should be 0 evseld <Configured evseld> startSchedule <startScheduleTime> numberPhases <Configured numberPhases> ChargingSchedule: duration 300 chargingRateUnit <Configured chargingRateUnit> startPeriod 0,60,120,180,260, limit 6,10,8,15,8 * <limit multiplier> Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier>	Note: set a TxProfile for 260 seconds: ChargingProfile 3: chargingProfilePurpose is <i>TxProfile</i> chargingProfileKind should be <i>Absolute</i> stackLevel should be 0 evseld <Configured evseld> startSchedule <startScheduleTime> numberPhases <Configured numberPhases> ChargingSchedule: duration 260 chargingRateUnit <Configured chargingRateUnit> startPeriod 0,50,140,200,240, limit 8,11,16,6,12 * <limit multiplier> Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier>
Reusable State(s): State is EnergyTransferStarted	

Main (Test scenario)	
Charging Station	CSMS

Main (Test scenario)	
2. The Charging Station responds with a GetCompositeScheduleResponse	<p>1. The Test System sends a GetCompositeScheduleRequest with evseld <Configured evseld> duration is 400 chargingRateUnit <Configured chargingRateUnit></p>

Tool validations
<p>* Step 2: (Message: GetCompositeScheduleResponse) status Accepted evseld <Configured evseld> ChargingSchedule: duration 400 chargingRateUnit <Configured chargingRateUnit> scheduleStart <The time the GetCompositeScheduleRequest was transmitted +/- <Configured max time deviation>> Note: The period of time between the <i>scheduleStart</i> from the SetChargingProfileRequest and the <i>scheduleStart</i> from the GetCompositeScheduleResponse is called x. startPeriod 0, limit $8 * \text{<limit multiplier>}$ (TxProfile) startPeriod (50 - x), limit $10 * \text{<limit multiplier>}$ (ChargingStationMaxProfile) startPeriod (200 - x), limit $6 * \text{<limit multiplier>}$ (TxProfile) startPeriod (240 - x), limit $10 * \text{<limit multiplier>}$ (ChargingStationMaxProfile) startPeriod (260 - x), limit $8 * \text{<limit multiplier>}$ (TxDefaultProfile) startPeriod (300 - x), limit $10 * \text{<limit multiplier>}$ (ChargingStationMaxProfile) Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></p>
<p>Post scenario validations: N/a</p>

TC_K_42_CS: Get Composite Schedule - chargingRateUnit not supported

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)	<ul style="list-style-type: none"> - The Charging Station does NOT support one of the chargingRateUnits; A or W. - The Test System 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 Test System sends a GetCompositeScheduleRequest with evseld 0 duration is 300 chargingRateUnit <Configured unsupported chargingRateUnit>

Tool validations
* Step 2: (Message: GetCompositeScheduleResponse) status <i>Rejected</i> schedule is omitted
Post scenario validations: N/a

TC_K_47_CS: Get Composite Schedule - Unknown EVSEId

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 evseld.
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 Test System sends a GetCompositeScheduleRequest with evseld <Configured number of evse> + 1 duration is 300 chargingRateUnit <Configured chargingRateUnit>

Tool validations
* Step 2: (Message: GetCompositeScheduleResponse) status <i>Rejected</i> schedule is omitted
Post scenario validations: N/a

TC_K_52_CS: EMS Control - Set / Update External Charging Limit (not on a transaction) - ChargingStationExternalConstraints in report

Test case name	EMS Control - Set / Update 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 <i>ChargingStatioExternalConstraints</i> .
Purpose	To verify if the charging station is able to correctly report an external charging limit as <i>ChargingStationExternalConstraints</i> .
Prerequisite(s)	n/a

Before (Preparations)
Configuration State: N/a
Memory State: An external charging limit has been submitted to Charging Station.
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a GetChargingProfilesResponse	1. The Test System sends a GetChargingProfilesRequest with chargingProfile.chargingProfilePurpose <i>ChargingStationExternalConstraints</i>
3. The Charging Station sends a ReportChargingProfilesRequest	4. The Test System responds with a ReportChargingProfilesResponse
<u>Note(s):</u> - If tbc is True at Step 3 then step 3 and 4 will be repeated	

Tool validations
<p>* Step 2: Message GetChargingProfilesResponse - status <i>Accepted</i></p> <p>* Step 3: Message ReportChargingProfilesRequest - requestId <i>Same id as in the request in step 1</i> - chargingProfile.chargingProfilePurpose <i>ChargingStationExternalConstraints</i></p>
Post scenario validations: - All report messages have been received and at least one <i>ChargingStationExternalConstraints</i> is returned.

TC_K_53_CS: Charging with load leveling based on High Level Communication - Success

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 <i>EVConnectedPreSession</i>

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>Authorized15118</i>	
2. The Charging Station sends a NotifyEVChargingNeedsRequest	3. The Test System responds with a NotifyEVChargingNeedsResponse with status <i>Accepted</i>
5. The Charging Station responds with a SetChargingProfileResponse	4. The Test System sends a SetChargingProfileRequest with chargingProfile.chargingProfilePurpose <i>TxProfile</i> chargingProfile.transactionId <i><transactionId></i> chargingProfile.chargingSchedule[0].id <i><Id generated by Test System></i> chargingProfile.chargingSchedule[0].chargingRateUnit <i><Configured chargingRateUnit></i> chargingProfile.chargingSchedule[0].chargingSchedulePeriod.startPeriod <i>0</i> chargingProfile.chargingSchedule[0].chargingSchedulePeriod.limit <i>6 * <limit multiplier></i> Note: Check <i>Determine Charging Profile Limit Multiplier</i> for <i><limit multiplier></i>
6. The Charging Station sends a NotifyEVChargingScheduleRequest	7. The Test System responds with a NotifyEVChargingScheduleResponse with status <i>Accepted</i>
8. The Charging Station sends a TransactionEventRequest .	9. The Test System responds with a TransactionEventResponse .

Tool validations

* Step 1:
Message: **TransactionEventRequest** of *Authorized15118* step 3
- **eventType** must be *Updated* or *Started*
- **triggerReason** must be *Authorized*

* Step 2:
Message: **NotifyEVChargingNeedsRequest**
IF **chargingNeeds.acChargingParameters** is <omitted> THEN
chargingNeeds.dcChargingParameters must be <not omitted>
END IF
IF **chargingNeeds.dcChargingParameters** is <omitted> THEN
chargingNeeds.acChargingParameters must be <not omitted>
END IF

* Step 5:
Message: **SetChargingProfileResponse**
- **status** must be *Accepted*

* Step 6:
Message: **NotifyEVChargingScheduleRequest**
- **chargingSchedule** must be within bounds of **chargingSchedule** of step 4

* Step 8:
Message: **TransactionEventRequest**
- **triggerReason** must be *ChargingStateChanged*
- **transactionInfo.chargingState** must be *Charging*

Post scenario validations:

N/a

TC_K_54_CS: Charging with load leveling based on High Level Communication - No SASchedule (rejected)

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 <i>EVConnectedPreSession</i>

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>Authorized15118</i>	
2. The Charging Station sends a NotifyEVChargingNeedsRequest .	3. The Test System responds with a NotifyEVChargingNeedsResponse . With status <i>Rejected</i>
4. The Charging Station sends a NotifyEVChargingScheduleRequest . <u>Note(s):</u> - <i>This step is optional. The Charging Station will only send it when the EV returns a charging profile.</i>	5. The Test System responds with a NotifyEVChargingScheduleResponse . With status <i>Accepted</i>
6. The Charging Station sends a TransactionEventRequest .	7. The Test System responds with a TransactionEventResponse .

Tool validations
<p>* Step 2: (Message: NotifyEVChargingNeedsRequest) evseld <Configured evseld></p> <p>* Step 4: (Message: NotifyEVChargingScheduleRequest) evseld <Configured evseld></p> <p>* Step 6: Message: TransactionEventRequest - triggerReason must be <i>ChargingStateChanged</i> - transactionInfo.chargingState must be <i>Charging</i></p>
Post scenario validations: N/a

TC_K_56_CS: Charging with load leveling based on High Level Communication - Offline

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)
<p>Configuration State: RetryBackOffWaitMinimum is <i><Configured RetryBackOffWaitMinimum></i></p> <p>Memory State: SetChargingProfile with ChargingProfile: chargingProfilePurpose is <i>TxDefaultProfile</i> chargingProfileKind should be <i>Absolute</i> stackLevel should be <i>0</i> evseld <i><Configured evseld></i> validFrom <i><current dateTime - <Configured max time deviation> seconds></i> validTo <i><current dateTime + <Configured max time deviation> + 401 seconds></i> startSchedule <i><current dateTime - <Configured max time deviation> seconds></i> numberPhases <i><Configured numberPhases></i></p> <p>ChargingSchedule: duration <i>400</i> chargingRateUnit <i><Configured chargingRateUnit></i> startPeriod <i>0</i>, limit <i>6 * <limit multiplier></i> Note: Check Determine Charging Profile Limit Multiplier for <i><limit multiplier></i></p> <p>Reusable State(s): State is EVConnectedPreSession State is Authorized15118</p>

Main (Test scenario)	
Charging Station	CSMS
	1. The Test System closes the WebSocket connection AND does not accept a reconnect.
	2. The Test System accepts the reconnection attempt from the Charging Station, after 90 seconds.
3. The Charging Station sends a NotifyEVChargingScheduleRequest . <u>Note(s):</u> - This step is optional. - It is allowed to execute this step either before or after the TransactionEventRequest from step 5.	4. The Test System responds with a NotifyEVChargingScheduleResponse . With status Accepted
5. The Charging Station sends a TransactionEventRequest .	6. The Test System responds with a TransactionEventResponse .

Tool validations
<div>* Step 3: (Message: NotifyEVChargingScheduleRequest) evseld <Configured evseld></div> <div>* Step 5: Message: TransactionEventRequest - triggerReason must be <i>ChargingStateChanged</i> - transactionInfo.chargingState must be <i>Charging</i> - offline <i>true</i></div>
<div>Post scenario validations: N/a</div>

TC_K_57_CS: Renegotiating a Charging Schedule ISO 15118-2 - Initiated by EV

Test case name	Renegotiating a Charging Schedule ISO 15118-2 - 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 <i>EVConnectedPreSession</i> State is <i>Authorized15118</i> State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>RenegotiateChargingLimits</i>	

Tool validations
N/a
Post scenario validations: N/a

TC_K_58_CS: Renegotiating a Charging Schedule ISO 15118-2 - Initiated by CSMS

Test case name	Renegotiating a Charging Schedule ISO 15118-2 - 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 <i>EVConnectedPreSession</i> State is <i>Authorized15118</i> State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetChargingProfileResponse	1. The Test System sends a SetChargingProfileRequest with chargingProfile.chargingProfilePurpose <i>TxProfile</i> chargingProfile.transactionId <i><Provided transactionId from before></i> chargingProfile.chargingSchedule[0].id <i><Id generated by Test System></i> chargingProfile.chargingSchedule[0].chargingRateUnit <i><Configured chargingRateUnit></i> chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].startPeriod <i>0</i> chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].limit <i>8 * <limit multiplier></i> <i>Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></i>
3. The Charging Station sends a NotifyEVChargingScheduleRequest . <u>Note(s):</u> - EV should send its calculated charging profile, but Charging Station has no control over it. If EV does not send it, Charging Station is recommended to return a schedule matching the provided charging schedule from CSMS.	4. The Test System responds with a NotifyEVChargingScheduleResponse . With status <i>Accepted</i>

Tool validations
<p>* Step 2: (Message: SetChargingProfileResponse) status <i>Accepted</i></p> <p>* Step 3: <i>Check that EV charging schedule matches or fits within charging profile</i> (Message: NotifyEVChargingScheduleRequest) evseld <i><Configured evseld></i> chargingSchedule.chargingSchedule[0].chargingRateUnit <i><Configured chargingRateUnit></i> chargingSchedule.chargingSchedule[0].chargingSchedulePeriod[0].startPeriod <i>0</i> chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].limit <i><= 8 * <limit multiplier></i> <i>Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></i></p>
<p>Post scenario validations: N/a</p>

TC_K_100_CS: Set Charging Profile - maxOfflineDuration

Test case name	Set Charging Profile - maxOfflineDuration
Test case Id	TC_K_100_CS
Use case Id(s)	K01
Requirement(s)	K01.FR.101, K01.FR.102, K01.FR.103
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.
Purpose	To verify if the Charging station stops using the charging profile when Charging station is longer offline than specified in the charging profile's maxOfflineDuration.
Prerequisite(s)	N/a

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetChargingProfileResponse	<p>1. The Test System sends a SetChargingProfileRequest with evseld 0 chargingProfile.id <Configured chargingProfileId> chargingProfile.stackLevel 0 chargingProfile.chargingProfilePurpose TxDefaultProfile chargingProfile.chargingSchedule.duration <omitted> chargingProfile.chargingSchedule.chargingRateUnit <Configured chargingRateUnit> chargingProfile.chargingSchedule.chargingSchedulePeriod[0].startPeriod 0 chargingProfile.chargingSchedule.chargingSchedulePeriod[0].limit 6 * <limit multiplier> Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier> chargingProfile.chargingSchedule.chargingSchedulePeriod[0].numberPhases <Configured numberPhases></p>

Main (Test scenario)	
4. The Charging Station responds with a SetChargingProfileResponse	<p>3. The Test System sends a SetChargingProfileRequest with evseld 0 chargingProfile.id <Configured chargingProfileId2> chargingProfile.stackLevel 1 chargingProfile.chargingProfilePurpose TxDefaultProfile chargingProfile.maxOfflineDuration <Configured maxOfflineDuration> chargingProfile.invalidAfterOfflineDuration true chargingProfile.chargingSchedule.duration <omitted> chargingProfile.chargingSchedule.chargingRateUnit <Configured chargingRateUnit> chargingProfile.chargingSchedule.chargingSchedulePeriod[0].startPeriod 0 chargingProfile.chargingSchedule.chargingSchedulePeriod[0].limit 7 * <limit multiplier> <i>Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></i> chargingProfile.chargingSchedule.chargingSchedulePeriod[0].numberPhases <Configured numberPhases></p>
5. Execute Reusable State EnergyTransferStarted	
6. The Test System closes the WebSocket connection AND does not accept a reconnect.	
7. The Test System immediately accepts a reconnection attempt from the Charging Station.	
9. The Charging Station responds with a GetCompositeScheduleResponse	<p>8. The Test System sends a GetCompositeScheduleRequest with evseld <Configured evseld> duration is 900 chargingRateUnit <Configured chargingRateUnit></p>
10. The Test System closes the WebSocket connection AND does not accept a reconnect.	
11. The Test System waits for the duration of <Configured maxOfflineDuration>.	
12. The Test System accepts reconnection attempt from the Charging Station.	
14. The Charging Station responds with a GetCompositeScheduleResponse	<p>13. The Test System sends a GetCompositeScheduleRequest with evseld <Configured evseld> duration is 900 chargingRateUnit <Configured chargingRateUnit></p>
16. The Charging Station responds with a SetChargingProfileResponse	<p>15. The Test System sends a SetChargingProfileRequest with evseld 0 chargingProfile.id <Configured chargingProfileId2> chargingProfile.stackLevel 1 chargingProfile.chargingProfilePurpose TxDefaultProfile chargingProfile.maxOfflineDuration <Configured maxOfflineDuration> chargingProfile.invalidAfterOfflineDuration false chargingProfile.chargingSchedule.chargingRateUnit <Configured chargingRateUnit> chargingProfile.chargingSchedule.chargingSchedulePeriod[0].startPeriod 0 chargingProfile.chargingSchedule.chargingSchedulePeriod[0].limit 8 * <limit multiplier> <i>Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></i> chargingProfile.chargingSchedule.chargingSchedulePeriod[0].numberPhases <Configured numberPhases></p>
17. The Test System closes the WebSocket connection AND does not accept a reconnect.	
18. The Test System waits for the duration of <Configured maxOfflineDuration>.	
19. The Test System accepts reconnection attempt from the Charging Station.	

Main (Test scenario)	
21. The Charging Station responds with a GetCompositeScheduleResponse	20. The Test System sends a GetCompositeScheduleRequest with evseld <Configured evseld> duration is 900 chargingRateUnit <Configured chargingRateUnit>
23. The Charging Station responds with a SetChargingProfileResponse	22. The Test System sends a SetChargingProfileRequest with evseld 0 chargingProfile.id <Configured chargingProfileId2> chargingProfile.stackLevel 1 chargingProfile.chargingProfilePurpose TxDefaultProfile chargingProfile.maxOfflineDuration 0 chargingProfile.invalidAfterOfflineDuration true chargingProfile.chargingSchedule.chargingRateUnit <Configured chargingRateUnit> chargingProfile.chargingSchedule.chargingSchedulePeriod[0].startPeriod 0 chargingProfile.chargingSchedule.chargingSchedulePeriod[0].limit 9 * <limit multiplier> Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier> chargingProfile.chargingSchedule.chargingSchedulePeriod[0].numberPhases <Configured numberPhases>
24. The Test System closes the WebSocket connection AND does not accept a reconnect.	
25. The Test System accepts reconnection attempt from the Charging Station.	
27. The Charging Station responds with a GetCompositeScheduleResponse	26. The Test System sends a GetCompositeScheduleRequest with evseld <Configured evseld> duration is 900 chargingRateUnit <Configured chargingRateUnit>

Tool validations

* Step 2:

Message **SetChargingProfileResponse**

- **status** must be *Accepted*

* Step 4:

Message **SetChargingProfileResponse**

- **status** must be *Accepted*

* Step 9:

Message: **GetCompositeScheduleResponse**

- **status** must be *Accepted*

- **schedule.chargingRateUnit** must be *<Configured chargingRateUnit>*

- **schedule.chargingSchedulePeriod[0].startPeriod** must be *0*

- **schedule.chargingSchedulePeriod[0].limit** must be *7 * <limit multiplier>*

Note: Check [Determine Charging Profile Limit Multiplier](#) for *<limit multiplier>*

* Step 14:

Message: **GetCompositeScheduleResponse**

- **status** must be *Accepted*

- **schedule.chargingRateUnit** must be *<Configured chargingRateUnit>*

- **schedule.chargingSchedulePeriod[0].startPeriod** must be *0*

- **schedule.chargingSchedulePeriod[0].limit** must be *6 * <limit multiplier>*

Note: Check [Determine Charging Profile Limit Multiplier](#) for *<limit multiplier>*

* Step 21:

Message: **GetCompositeScheduleResponse**

- **status** must be *Accepted*

- **schedule.chargingRateUnit** must be *<Configured chargingRateUnit>*

- **schedule.chargingSchedulePeriod[0].startPeriod** must be *0*

- **schedule.chargingSchedulePeriod[0].limit** must be *8 * <limit multiplier>*

Note: Check [Determine Charging Profile Limit Multiplier](#) for *<limit multiplier>*

* Step 27:

Message: **GetCompositeScheduleResponse**

- **status** must be *Accepted*

- **schedule.chargingRateUnit** must be *<Configured chargingRateUnit>*

- **schedule.chargingSchedulePeriod[0].startPeriod** must be *0*

- **schedule.chargingSchedulePeriod[0].limit** must be *6 * <limit multiplier>*

Note: Check [Determine Charging Profile Limit Multiplier](#) for *<limit multiplier>*

Post scenario validations:

N/a

TC_K_101_CS: Set Charging Profile - Change operation mode

Test case name	Set Charging Profile - Change operation mode
Test case Id	TC_K_101_CS
Use case Id(s)	K01
Requirement(s)	K01.FR.110
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.
Purpose	To verify if the Charging station sends an update to the CSMS when a charging profile changes operationMode.
Prerequisite(s)	V2XChargingCtrlr.SupportedOperationModes contains multiple values

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetChargingProfileResponse	1. The Test System sends a SetChargingProfileRequest with evseld 0 chargingProfile.id <Configured chargingProfileId> chargingProfile.stackLevel 0 chargingProfile.chargingProfilePurpose TxDefaultProfile chargingProfile.chargingProfileKind Relative chargingProfile.chargingSchedule.chargingRateUnit <Configured chargingRateUnit> chargingProfile.chargingSchedule.chargingSchedulePeriod[0].s tartPeriod 0 chargingProfile.chargingSchedule.chargingSchedulePeriod[0].n umberPhases <Configured numberPhases> chargingProfile.chargingSchedule.chargingSchedulePeriod[0].o perationMode ChargingOnly chargingProfile.chargingSchedule.chargingSchedulePeriod[1].s tartPeriod 10 chargingProfile.chargingSchedule.chargingSchedulePeriod[1].n umberPhases <Configured numberPhases> chargingProfile.chargingSchedule.chargingSchedulePeriod[1].o perationMode <Configured operationMode> (not ChargingOnly)
3. Execute Reusable State <i>EnergyTransferStarted</i>	
4. The Charging Station sends a TransactionEventRequest	5. The Test System responds with a TransactionEventResponse

Tool validations
<p>* Step 2:</p> <p>Message SetChargingProfileResponse</p> <ul style="list-style-type: none">- status must be <i>Accepted</i> <p>* Step 4:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none">- eventType must be <i>Updated</i>- triggerReason must be <i>OperationModeChanged</i>- transactionInfo.operationMode must be <i><Configured operationMode></i> (not <i>ChargingOnly</i>)
<p>Post scenario validations:</p> <p>N/a</p>

TC_K_102_CS: Set Charging Profile - limitAtSoc

Test case name	Set Charging Profile - limitAtSoc
Test case Id	TC_K_102_CS
Use case Id(s)	K01
Requirement(s)	K01.FR.100
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.
Purpose	To verify if the Charging station caps the charging speed when reaching specified <code>limitAtSoc</code> .
Prerequisite(s)	- Charging station supports receiving SoC from the EV. e.g. by CHaDeMo, ISO15118-2 or ISO15118-20 - SmartChargingCtrlr.SupportsFeature[LimitAtSoc] is <i>true</i>

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetChargingProfileResponse	<p>1. The Test System sends a SetChargingProfileRequest with evseld 0</p> <p>chargingProfile.id <Configured chargingProfileId></p> <p>chargingProfile.chargingProfilePurpose TxDefaultProfile</p> <p>chargingProfile.chargingSchedule.chargingRateUnit <Configured chargingRateUnit></p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod[0].startPeriod 0</p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod[0].limit 8 * <limit multiplier></p> <p>Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod[0].numberPhases <Configured numberPhases></p> <p>chargingProfile.chargingSchedule.limitAtSoc.soc <Configured limitAtSocSoc></p> <p>chargingProfile.chargingSchedule.limitAtSoc.limit 6 * <limit multiplier></p> <p>Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></p>
3. Execute Reusable State EnergyTransferStarted	
<u>Manual Action:</u> Use EV to send out a SoC value less than <Configured limitAtSocSoc>	
5. The Charging Station responds with a GetCompositeScheduleResponse	<p>4. The Test System sends a GetCompositeScheduleRequest with evseld <Configured evseld></p> <p>duration is 900</p> <p>chargingRateUnit <Configured chargingRateUnit></p>
<u>Manual Action:</u> Use EV to send out a SoC value greater or equal to <Configured limitAtSocSoc>_	
7. The Charging Station responds with a GetCompositeScheduleResponse	<p>6. The Test System sends a GetCompositeScheduleRequest with evseld <Configured evseld></p> <p>duration is 900</p> <p>chargingRateUnit <Configured chargingRateUnit></p>

Main (Test scenario)	
<u>Manual Action:</u> Use EV to send out a SoC value less than <Configured limitAtSocSoc>	
9. The Charging Station responds with a GetCompositeScheduleResponse	8. The Test System sends a GetCompositeScheduleRequest with evseld <Configured evseld> duration is 900 chargingRateUnit <Configured chargingRateUnit>
Tool validations	
<p>* Step 2: Message: SetChargingProfileResponse - status must be <i>Accepted</i></p> <p>* Step 5: Message: GetCompositeScheduleResponse - status must be <i>Accepted</i> - schedule.chargingRateUnit must be <Configured chargingRateUnit> - schedule.chargingSchedulePeriod[0].startPeriod must be 0 - schedule.chargingSchedulePeriod[0].limit must be $8 * \text{<limit multiplier>}$ Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></p> <p>* Step 7: Message: GetCompositeScheduleResponse - status must be <i>Accepted</i> - schedule.chargingRateUnit must be <Configured chargingRateUnit> - schedule.chargingSchedulePeriod[0].startPeriod must be 0 - schedule.chargingSchedulePeriod[0].limit must be $6 * \text{<limit multiplier>}$ Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></p> <p>* Step 9: Message: GetCompositeScheduleResponse - status must be <i>Accepted</i> - schedule.chargingRateUnit must be <Configured chargingRateUnit> - schedule.chargingSchedulePeriod[0].startPeriod must be 0 - schedule.chargingSchedulePeriod[0].limit must be $8 * \text{<limit multiplier>}$ Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></p>	
Post scenario validations: N/a	

TC_K_103_CS: Set Charging Profile - Local time - TimeOffset

Test case name	Set Charging Profile - Local time - TimeOffset
Test case Id	TC_K_103_CS
Use case Id(s)	K01
Requirement(s)	K01.FR.90
System under test	Charging Station
Description	To enable the CSMS to set charging profiles with local times so the charging power can be influences on local time base independent on DST.
Purpose	To verify if the Charging station is able to calculate the Zulu time from a chargingProfileSchedule using local times.
Prerequisite(s)	- SmartChargingCtrlr.SupportsFeature[UseLocalTime] is true - ClockCtrlr.TimeOffset is present

Before (Preparations)
Configuration State: - ClockCtrlr.TimeZone is "" (if present)
Memory State: N/a
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with SetVariablesResponse	1. Test System sends SetVariablesRequest with: - setVariableData[0].variable.name = <i>TimeOffset</i> - setVariableData[0].component.name = <i>ClockCtrlr</i> - setVariableData[0].attributeValue = <i>+02:00</i> - setVariableData[0].attributeType = <i>Actual</i> - setVariableData[1].variable.name = <i>TimeZone</i> - setVariableData[1].component.name = <i>ClockCtrlr</i> - setVariableData[1].attributeValue = "" - setVariableData[1].attributeType = <i>Actual</i>
4. The Charging Station responds with a SetChargingProfileResponse	3. The Test System sends a SetChargingProfileRequest with evseld 0 chargingProfile.id <Configured chargingProfileId> chargingProfile.chargingProfilePurpose <i>TxDefaultProfile</i> chargingProfile.chargingSchedule.useLocalTime <i>true</i> chargingProfile.chargingSchedule.duration <Configured duration> chargingProfile.chargingSchedule.startSchedule <current dateTime + 02:10> chargingProfile.chargingSchedule.chargingRateUnit <Configured chargingRateUnit> chargingProfile.chargingSchedule.chargingSchedulePeriod.startPeriod 0 chargingProfile.chargingSchedule.chargingSchedulePeriod.limit 6 * <limit multiplier> <i>Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></i> chargingProfile.chargingSchedule.chargingSchedulePeriod.numberPhases <Configured numberPhases>

Main (Test scenario)	
6. The Charging Station responds with a GetCompositeScheduleResponse	5. The Test System sends a GetCompositeScheduleRequest with evseld 0 duration is 900 chargingRateUnit <Configured chargingRateUnit>

Tool validations
<p>* Step 2: Message: SetVariablesResponse</p> <ul style="list-style-type: none"> - setVariableResult[0].attributeStatus must be <i>Accepted</i> - setVariableResult[0].component.name = <i>ClockCtrlr</i> - setVariableResult[0].variable.name = <i>TimeOffset</i> <p>* Step 4: Message SetChargingProfileResponse</p> <ul style="list-style-type: none"> - status must be <i>Accepted</i> <p>* Step 6: Message: GetCompositeScheduleResponse Note: The period of time between the <i>scheduleStart</i> from the <i>SetChargingProfileRequest</i> and the <i>scheduleStart</i> from the <i>GetCompositeScheduleResponse</i> is called <i>x</i>.</p> <ul style="list-style-type: none"> - status must be <i>Accepted</i> - schedule.evseld must be 0 - schedule.scheduleStart must be <current dateTime +/- max deviation> - schedule.chargingSchedulePeriod[0].startPeriod 0 - schedule.chargingSchedulePeriod[1].startPeriod 600 - x - schedule.chargingSchedulePeriod[1].limit 6 * <limit multiplier> <p>Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></p>
<p>Post scenario validations: N/a</p>

The following validation is incorrect: schedule.scheduleStart must be <current dateTime + 00:10>

The requested composite schedule is always the current dateTime.

But the first period will be the max local limit and the second period starts at 10 seconds with limit 6. <<<

TC_K_104_CS: Set Charging Profile - PriorityCharging

Test case name	Set Charging Profile - PriorityCharging
Test case Id	TC_K_104_CS
Use case Id(s)	K01
Requirement(s)	K01.FR.70, K01.FR.71
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 validate a chargingProfile of purpose <i>PriorityCharging</i> .
Prerequisite(s)	SmartChargingCtrlr.SupportedAdditionalPurposes contains <i>PriorityCharging</i>

Before (Preparations)
<p>Configuration State: N/a</p>
<p>Memory State: N/a</p>

Before (Preparations)	
Reusable State: N/a	
Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetChargingProfileResponse	1. The Test System sends a SetChargingProfileRequest with evseld 0 chargingProfile.id <Configured chargingProfileId> chargingProfile.chargingProfilePurpose PriorityCharging chargingProfile.chargingSchedule.duration <Configured duration> chargingProfile.chargingSchedule.chargingRateUnit <Configured chargingRateUnit> chargingProfile.chargingSchedule.chargingSchedulePeriod.startPeriod 0 chargingProfile.chargingSchedule.chargingSchedulePeriod.limit 6 * <limit multiplier> Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier> chargingProfile.chargingSchedule.chargingSchedulePeriod.numberPhases <Configured numberPhases>
4. The Charging Station responds with a SetChargingProfileResponse	3. The Test System sends a SetChargingProfileRequest with evseld <Configured evseld> chargingProfile.id <Configured chargingProfileId2> chargingProfile.chargingProfilePurpose PriorityCharging chargingProfile.chargingSchedule.chargingRateUnit <Configured chargingRateUnit> chargingProfile.chargingSchedule.chargingSchedulePeriod.startPeriod 0 chargingProfile.chargingSchedule.chargingSchedulePeriod.limit 6 * <limit multiplier> Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier> chargingProfile.chargingSchedule.chargingSchedulePeriod.numberPhases <Configured numberPhases> chargingProfile.chargingSchedule.chargingSchedulePeriod.operationMode <Configured operationMode other than ChargingOnly>
Tool validations	
* Step 2: Message SetChargingProfileResponse - status must be <i>Rejected</i> - statusInfo.reasonCode must be <i>InvalidSchedule</i> or <omitted>	
* Step 4: Message SetChargingProfileResponse - status must be <i>Rejected</i> - statusInfo.reasonCode must be <i>InvalidSchedule</i>	
Post scenario validations: n/a	

TC_K_105_CS: Set Charging Profile - ChargingStationMaxProfile persistent over reboot

Test case name	Set Charging Profile - ChargingStationMaxProfile persistent over reboot
Test case Id	TC_K_105_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 of purpose ChargingStationMaxProfile persistent over reboot as described at the OCPP specification.
Prerequisite(s)	N/a

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetChargingProfileResponse	<p>1. The Test System sends a SetChargingProfileRequest with evseld 0 chargingProfile.id <Configured chargingProfileId> chargingProfile.chargingProfilePurpose <i>ChargingStationMaxProfile</i> chargingProfile.chargingSchedule.duration <Configured duration> chargingProfile.chargingSchedule.chargingRateUnit <Configured chargingRateUnit> chargingProfile.chargingSchedule.chargingSchedulePeriod.startPeriod 0 chargingProfile.chargingSchedule.chargingSchedulePeriod.limit 6 * <limit multiplier> <i>Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></i> chargingProfile.chargingSchedule.chargingSchedulePeriod.numberPhases <Configured numberPhases></p>
3. Execute Reusable State <i>Booted</i>	
5. The Charging Station responds with a GetChargingProfilesResponse	<p>4. The Test System sends a GetChargingProfilesRequest with chargingProfile.chargingProfileId[0] <Configured chargingProfileId> chargingProfile.chargingProfileId[1] <Configured chargingProfileId2></p>
6. The Charging Station sends a ReportChargingProfilesRequest	<p>7. The Test System responds with a ReportChargingProfilesResponse</p>
<p><u>Note(s):</u> - If tbc is True at Step 6 then step 6 and 7 will be repeated</p>	

Tool validations
<p>* Step 2:</p> <p>Message SetChargingProfileResponse</p> <ul style="list-style-type: none">- status must be <i>Accepted</i> <p>* Step 5:</p> <p>Message GetChargingProfilesResponse</p> <ul style="list-style-type: none">- status must be <i>Accepted</i> <p>* Step 6:</p> <p>Message ReportChargingProfilesRequest</p> <ul style="list-style-type: none">- requestId must be <i>Same Id as in the GetChargingProfilesRequest in step 4</i>- EVSEId must be <i>0</i>- chargingProfile must be <i><Configured chargingProfile1></i>
<p>Post scenario validations:</p> <ul style="list-style-type: none">- The same profile is reported as send in step 1

TC_K_106_CS: Set Charging Profile - randomizedDelay

Test case name	Set Charging Profile - randomizedDelay
Test case Id	TC_K_106_CS
Use case Id(s)	K01
Requirement(s)	K01.FR.92, K01.FR.93
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 with randomised delays, to avoid sharp peaks in the grid.
Purpose	To verify if the Charging station is able to execute a chargingProfile for which randomizedDelay is set.
Prerequisite(s)	SmartChargingCtrlr.SupportsFeature[RandomizedDelay] is <i>true</i>

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>EnergyTransferStarted</i>	

Main (Test scenario)	
<p>3. The Charging Station responds with a SetChargingProfileResponse</p>	<p>2. The Test System sends a SetChargingProfileRequest with evseld 0</p> <p>chargingProfile.id <Configured chargingProfileId> chargingProfile.chargingProfilePurpose TxDefaultProfile chargingProfile.chargingProfileKind Recurring chargingProfile.recurrencyKind Daily chargingProfile.chargingSchedule.chargingRateUnit <Configured chargingRateUnit> chargingProfile.chargingSchedule.randomizedDelay 5760 chargingProfile.chargingSchedule.duration 86400 chargingProfile.chargingSchedule.startSchedule <Current dateTime> chargingProfile.chargingSchedule.chargingSchedulePeriod[0].startPeriod 0 chargingProfile.chargingSchedule.chargingSchedulePeriod[0].limit 6 * <limit multiplier> <i>Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></i> chargingProfile.chargingSchedule.chargingSchedulePeriod[0].numberPhases <Configured numberPhases> chargingProfile.chargingSchedule.chargingSchedulePeriod[1].startPeriod 1 chargingProfile.chargingSchedule.chargingSchedulePeriod[1].limit 0 chargingProfile.chargingSchedule.chargingSchedulePeriod[1].numberPhases <Configured numberPhases> chargingProfile.chargingSchedule.chargingSchedulePeriod[2].startPeriod 28800 chargingProfile.chargingSchedule.chargingSchedulePeriod[2].limit 6 * <limit multiplier> <i>Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></i> chargingProfile.chargingSchedule.chargingSchedulePeriod[2].numberPhases <Configured numberPhases></p>
<p><u>Note(s):</u> To make the daily recurring schedule work while honouring the randomizedDelay, the chargingSchedulePeriod[0] must be the same as the last chargingSchedulePeriod[2].</p>	
<p>5. The Charging Station responds with a GetCompositeScheduleResponse</p>	<p>4. The Test System sends a GetCompositeScheduleRequest with evseld <Configured evseld> duration is 134400 chargingRateUnit <Configured chargingRateUnit></p>

Main (Test scenario)	
<p>7. The Charging Station responds with a SetChargingProfileResponse</p>	<p>6. The Test System sends a SetChargingProfileRequest with evseld <Configured evseld> chargingProfile.id <Configured chargingProfileId> chargingProfile.chargingProfilePurpose TxProfile chargingProfile.transactionId <transactionId provided by the Charging Station in TransactionEventRequest> chargingProfile.chargingSchedule.chargingRateUnit <Configured chargingRateUnit> chargingProfile.chargingSchedule.duration 38400 chargingProfile.chargingSchedule.randomizedDelay 5760 chargingProfile.chargingSchedule.chargingSchedulePeriod[0].startPeriod 0 chargingProfile.chargingSchedule.chargingSchedulePeriod[0].limit 3 * <limit multiplier> Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier> chargingProfile.chargingSchedule.chargingSchedulePeriod[0].numberPhases <Configured numberPhases> chargingProfile.chargingSchedule.chargingSchedulePeriod[1].startPeriod 9600 chargingProfile.chargingSchedule.chargingSchedulePeriod[1].limit 0 chargingProfile.chargingSchedule.chargingSchedulePeriod[1].numberPhases <Configured numberPhases> chargingProfile.chargingSchedule.chargingSchedulePeriod[2].startPeriod 19200 chargingProfile.chargingSchedule.chargingSchedulePeriod[2].limit 2 * <limit multiplier> Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier> chargingProfile.chargingSchedule.chargingSchedulePeriod[2].numberPhases <Configured numberPhases></p>
<p>9. The Charging Station responds with a GetCompositeScheduleResponse</p>	<p>8. The Test System sends a GetCompositeScheduleRequest with evseld <Configured evseld> duration is 134400 chargingRateUnit <Configured chargingRateUnit></p>

Tool validations

* Step 3:

Message **SetChargingProfileResponse**

- **status** must be *Accepted*

* Step 5:

*Note: The period of time between sending the **SetChargingProfileRequest** in step 2 and the **scheduleStart** from the **GetCompositeScheduleResponse** is called **x**. The first period lasts between **1** and **1 + <random>** seconds, where **<random>** is between 0 and **randomizedDelay**. If **x > 1 + <random>** then the first period will be missed in the composite schedule.*

Message: **GetCompositeScheduleResponse**

- **status** must be *Accepted*

- **schedule.duration** must be 134400

- **schedule.chargingSchedulePeriod[0].startPeriod** must be 0

- **schedule.chargingSchedulePeriod[0].limit** must be 6

- **schedule.chargingSchedulePeriod[1].startPeriod** must be between 1 and 5760 - **<x>**

- **schedule.chargingSchedulePeriod[1].limit** must be 0

- **schedule.chargingSchedulePeriod[2].startPeriod** must be between 28800 - **<x>** and 34560 - **<x>**

- **schedule.chargingSchedulePeriod[3].limit** must be 6 * **<limit multiplier>**

*Note: Check [Determine Charging Profile Limit Multiplier](#) for **<limit multiplier>***

- **schedule.chargingSchedulePeriod[4].startPeriod** must be between 86400 - **<x>**

- **schedule.chargingSchedulePeriod[4].limit** must be 0

- **schedule.chargingSchedulePeriod[5].startPeriod** must be between 115200 - **<x>** and 120960 - **<x>**

- **schedule.chargingSchedulePeriod[5].limit** must be 6 * **<limit multiplier>**

*Note: Check [Determine Charging Profile Limit Multiplier](#) for **<limit multiplier>***

* Step 9:

*Note: The period of time between sending the **SetChargingProfileRequest** in step 6 and the **scheduleStart** from the **GetCompositeScheduleResponse** is called **x**:*

Message: **GetCompositeScheduleResponse**

- **status** must be *Accepted*

- **schedule.duration** must be 134400

- **schedule.chargingSchedulePeriod[0].startPeriod** must be 0

- **schedule.chargingSchedulePeriod[0].limit** must be 3 * **<limit multiplier>**

*Note: Check [Determine Charging Profile Limit Multiplier](#) for **<limit multiplier>***

- **schedule.chargingSchedulePeriod[1].startPeriod** must be between 9600 - **<x>** and 15360 - **<x>**

- **schedule.chargingSchedulePeriod[1].limit** must be 0

- **schedule.chargingSchedulePeriod[2].startPeriod** must be between 19200 - **<x>** and 24960 - **<x>**

- **schedule.chargingSchedulePeriod[2].limit** must be 2 * **<limit multiplier>**

*Note: Check [Determine Charging Profile Limit Multiplier](#) for **<limit multiplier>***

- **schedule.chargingSchedulePeriod[3].startPeriod** must be 38400 - **<x>**

- **schedule.chargingSchedulePeriod[3].limit** must be 6 * **<limit multiplier>**

*Note: Check [Determine Charging Profile Limit Multiplier](#) for **<limit multiplier>***

- **schedule.chargingSchedulePeriod[4].startPeriod** must be 86400 - **<x>**

- **schedule.chargingSchedulePeriod[4].limit** must be 0

- **schedule.chargingSchedulePeriod[5].startPeriod** must be between 115200 - **<x>** and 120960 - **<x>**

- **schedule.chargingSchedulePeriod[5].limit** must be 6 * **<limit multiplier>**

*Note: Check [Determine Charging Profile Limit Multiplier](#) for **<limit multiplier>***

Post scenario validations:

N/a

TC_K_107_CS: Set Charging Profile - randomizedDelay - validations

Test case name	Set Charging Profile - randomizedDelay - validations
Test case Id	TC_K_107_CS
Use case Id(s)	K01
Requirement(s)	K01.FR.95
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 with randomised delays, to avoid sharp peaks in the grid.
Purpose	To verify if the Charging station is able to validate a chargingProfile which randomizedDelay is set.
Prerequisite(s)	SmartChargingCtrlr.SupportsFeature[RandomizedDelay] is true

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetChargingProfileResponse	1. The Test System sends a SetChargingProfileRequest with chargingProfile.chargingProfilePurpose ChargingStationMaxProfile chargingProfile.chargingSchedule.randomizedDelay 900
4. The Charging Station responds with a SetChargingProfileResponse	3. The Test System sends a SetChargingProfileRequest with chargingProfile.chargingProfilePurpose ChargingStationMaxProfile chargingProfile.chargingSchedule.randomizedDelay 0
6. The Charging Station responds with a SetChargingProfileResponse	5. The Test System sends a SetChargingProfileRequest with chargingProfile.chargingProfilePurpose PriorityCharging chargingProfile.chargingSchedule.randomizedDelay 900
8. The Charging Station responds with a SetChargingProfileResponse	7. The Test System sends a SetChargingProfileRequest with chargingProfile.chargingProfilePurpose PriorityCharging chargingProfile.chargingSchedule.randomizedDelay 0
10. The Charging Station responds with a SetChargingProfileResponse	9. The Test System sends a SetChargingProfileRequest with chargingProfile.chargingProfilePurpose LocalGeneration chargingProfile.chargingSchedule.randomizedDelay 900
12. The Charging Station responds with a SetChargingProfileResponse	11. The Test System sends a SetChargingProfileRequest with chargingProfile.chargingProfilePurpose LocalGeneration chargingProfile.chargingSchedule.randomizedDelay 0

Tool validations
<p>* Step 2:</p> <p>Message SetChargingProfileResponse</p> <ul style="list-style-type: none">- status must be <i>Rejected</i>- statusInfo.reasonCode must be <i>InvalidSchedule</i> <p>* Step 4:</p> <p>Message SetChargingProfileResponse</p> <ul style="list-style-type: none">- status must be <i>Accepted</i> <p>* Step 6:</p> <p>Message SetChargingProfileResponse</p> <ul style="list-style-type: none">- status must be <i>Rejected</i>- statusInfo.reasonCode must be <i>InvalidSchedule</i> <p>* Step 8:</p> <p>Message SetChargingProfileResponse</p> <ul style="list-style-type: none">- status must be <i>Accepted</i> <p>* Step 10:</p> <p>Message SetChargingProfileResponse</p> <ul style="list-style-type: none">- status must be <i>Rejected</i>- statusInfo.reasonCode must be <i>InvalidSchedule</i> <p>* Step 12:</p> <p>Message SetChargingProfileResponse</p> <ul style="list-style-type: none">- status must be <i>Accepted</i>
<p>Post scenario validations:</p> <p>N/a</p>

TC_K_108_CS: Set Charging Profile - randomizedDelay - random for each tx

Test case name	Set Charging Profile - randomizedDelay - random for each tx
Test case Id	TC_K_108_CS
Use case Id(s)	K01
Requirement(s)	K01.FR.92, K01.FR.93
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 with randomised delays, to avoid sharp peaks in the grid.
Purpose	To verify if the Charging station does calculate different random delays for each transaction.
Prerequisite(s)	SmartChargingCtrlr.SupportsFeature[RandomizedDelay] is true

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetChargingProfileResponse	<p>1. The Test System sends a SetChargingProfileRequest with evseld 0 chargingProfile.id <Configured chargingProfileId> chargingProfile.chargingProfilePurpose TxDefaultProfile chargingProfile.chargingProfileKind Recurring chargingProfile.recurrencyKind Daily chargingProfile.chargingSchedule.chargingRateUnit <Configured chargingRateUnit> chargingProfile.chargingSchedule.randomizedDelay 5760 chargingProfile.chargingSchedule.duration 86400 chargingProfile.chargingSchedule.startSchedule <Current dateTime> chargingProfile.chargingSchedule.chargingSchedulePeriod[0].startPeriod 0 chargingProfile.chargingSchedule.chargingSchedulePeriod[0].limit 0 chargingProfile.chargingSchedule.chargingSchedulePeriod[0].numberPhases <Configured numberPhases> chargingProfile.chargingSchedule.chargingSchedulePeriod[1].startPeriod 28800 chargingProfile.chargingSchedule.chargingSchedulePeriod[1].limit 6 * <limit multiplier> Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier> chargingProfile.chargingSchedule.chargingSchedulePeriod[1].numberPhases <Configured numberPhases></p>
3. Execute Reusable State EnergyTransferStarted	

Main (Test scenario)	
5. The Charging Station responds with a GetCompositeScheduleResponse	4. The Test System sends a GetCompositeScheduleRequest with evseld <Configured evseld> duration is 129600 chargingRateUnit <Configured chargingRateUnit>
<u>Note</u> : Stopping transaction	
6. Execute Reusable State <i>StopAuthorized</i>	
7. Execute Reusable State <i>EVConnectedPostSession</i>	
8. Execute Reusable State <i>EVDisconnected</i>	
9. Execute Reusable State <i>ParkingBayUnoccupied</i>	
<u>Note</u> : Restarting transaction	
10. Execute Reusable State <i>EnergyTransferStarted</i>	
12. The Charging Station responds with a GetCompositeScheduleResponse	11. The Test System sends a GetCompositeScheduleRequest with evseld <Configured evseld> duration is 134400 chargingRateUnit <Configured chargingRateUnit>
Tool validations	
<p>* Step 3: Message SetChargingProfileResponse - status must be Accepted</p> <p>* Step 12: <i>Note: The period of time between the scheduleStart from the GetCompositeScheduleResponse in step 5 and the scheduleStart from the GetCompositeScheduleResponse in step 12 is called x:</i> Message: GetCompositeScheduleResponse - status must be Accepted - schedule.duration must be 129600 - schedule.chargingSchedulePeriod[0].startPeriod must be 0 - schedule.chargingSchedulePeriod[1].startPeriod must NOT be within the range of <chargingSchedulePeriod[1].startPeriod of step 5> - x - 1 to <chargingSchedulePeriod[1].startPeriod of step 5> - x + 1</p>	
Post scenario validations: N/a	

TC_K_109_CS: EMS Control - Set Charging Profile - MaxExternalConstraintsId

Test case name	EMS Control - Set Charging Profile - MaxExternalConstraintsId
Test case Id	TC_K_109_CS
Use case Id(s)	K01
Requirement(s)	
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.
Purpose	To verify if the Charging station validates Charging Profile ids being set when <i>MaxExternalConstraintsId</i> is configured.
Prerequisite(s)	<ul style="list-style-type: none"> - Charging station supports Component Variable SmartChargingCtrlr.MaxExternalConstraintsId. - Charging station itself generates a Charging Profile of with chargingProfilePurpose <i>ChargingStationExternalConstraints</i>.

Before (Preparations)
Configuration State: SmartChargingCtrlr.MaxExternalConstraintsId is 10
Memory State: N/a
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a GetChargingProfilesResponse	1. The Test System sends a GetChargingProfilesRequest with chargingProfile.chargingProfilePurpose <i>ChargingStationExternalConstraints</i> requestId <Generated requestId>
3. The Charging Station sends a ReportChargingProfilesRequest	4. The Test System responds with a ReportChargingProfilesResponse
<u>Note(s):</u> - If tbc is True at Step 3 then step 3 and 4 will be repeated	

Tool validations
* Step 2: Message GetChargingProfilesResponse - status must be <i>Accepted</i> * Step 3: Message ReportChargingProfilesRequest - requestId must be <Generated requestId> - chargingProfile[*].id must be <= 10 - chargingProfile[*].chargingProfilePurpose must be <i>ChargingStationExternalConstraints</i>
Post scenario validations: N/a

TC_K_110_CS: EMS Control - Set Charging Profile - MaxExternalConstraintsId - validations

Test case name	EMS Control - Set Charging Profile - MaxExternalConstraintsId - validations
Test case Id	TC_K_110_CS
Use case Id(s)	K01
Requirement(s)	K01.FR.81
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.
Purpose	To verify if the Charging station validates Charging Profile ids being set when <i>MaxExternalConstraintsId</i> is configured.
Prerequisite(s)	Charging station supports Component Variable SmartChargingCtrlr.MaxExternalConstraintsId .

Before (Preparations)
Configuration State: SmartChargingCtrlr.MaxExternalConstraintsId is 99
Memory State: N/a
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetChargingProfileResponse	1. The Test System sends a SetChargingProfileRequest with chargingProfile.id 99
4. The Charging Station responds with a SetChargingProfileResponse	3. The Test System sends a SetChargingProfileRequest with chargingProfile.id 50
6. The Charging Station responds with a SetChargingProfileResponse	5. The Test System sends a SetChargingProfileRequest with chargingProfile.id 100

Tool validations
<p>* Step 2:</p> <p>Message SetChargingProfileResponse</p> <ul style="list-style-type: none"> - status must be <i>Rejected</i> - statusInfo.reasonCode must be <i>InvalidProfileId</i> <p>* Step 4:</p> <p>Message SetChargingProfileResponse</p> <ul style="list-style-type: none"> - status must be <i>Rejected</i> - statusInfo.reasonCode must be <i>InvalidProfileId</i> <p>* Step 6:</p> <p>Message SetChargingProfileResponse</p> <ul style="list-style-type: none"> - status must be <i>Accepted</i> <p>Post scenario validations: N/a</p>

TC_K_112_CS: Get Composite Schedule - randomizedDelay

Test case name	Get Composite Schedule - randomizedDelay
Test case Id	TC_K_112_CS
Use case Id(s)	K08
Requirement(s)	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 doesn't use the <code>randomizedDelay</code> when calculating a composite charging schedule while no transaction is running.
Prerequisite(s)	Charging Station has <code>SmartChargingCtrlr.SupportsFeature[RandomizedDelay] = true</code>

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetChargingProfileResponse	<p>1. The Test System sends a SetChargingProfileRequest with</p> <p>evseld 0</p> <p>chargingProfile.id <Configured chargingProfileId></p> <p>chargingProfile.stackLevel 0</p> <p>chargingProfile.chargingProfilePurpose TxDefaultProfile</p> <p>chargingProfile.chargingSchedule.chargingRateUnit <Configured chargingRateUnit></p> <p>chargingProfile.chargingSchedule.startSchedule <Current dateTime></p> <p>chargingProfile.chargingSchedule.randomizedDelay 1000</p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod[0].startPeriod 0</p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod[0].limit 6 * <limit multiplier></p> <p>Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod[0].numberPhases <Configured numberPhases></p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod[1].startPeriod 3600</p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod[1].limit 7 * <limit multiplier></p> <p>Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod[1].numberPhases <Configured numberPhases></p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod[2].startPeriod 7200</p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod[2].limit 8 * <limit multiplier></p> <p>Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod[2].numberPhases <Configured numberPhases></p>

Main (Test scenario)	
4. The Charging Station responds with a GetCompositeScheduleResponse	3. The Test System sends a GetCompositeScheduleRequest with evseld <Configured evseld> duration is 8000 chargingRateUnit <Configured chargingRateUnit>

Tool validations
<p>* Step 2: Message SetChargingProfileResponse - status must be Accepted</p> <p>* Step 4: Note: The period of time between sending the SetChargingProfileRequest in step 1 and the scheduleStart from the GetCompositeScheduleResponse is called x: Message: GetCompositeScheduleResponse - status must be Accepted - schedule.duration must be 8000 - schedule.chargingSchedulePeriod[0].startPeriod must be 0 - schedule.chargingSchedulePeriod[0].limit must be 6 * <limit multiplier> Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></p> <p>- schedule.chargingSchedulePeriod[1].startPeriod must be 3600 - <x> - schedule.chargingSchedulePeriod[1].limit must be 7 * <limit multiplier> Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></p> <p>- schedule.chargingSchedulePeriod[2].startPeriod must be 7200 - <x> - schedule.chargingSchedulePeriod[2].limit must be 8 * <limit multiplier> Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></p>
<p>Post scenario validations: N/a</p>

TC_K_113_CS: Renegotiating a Charging Schedule ISO 15118-20 - Initiated by CSMS

Test case name	Renegotiating a Charging Schedule ISO 15118-20 - Initiated by CSMS
Test case Id	TC_K_113_CS
Use case Id(s)	K16
Requirement(s)	K16.FR.01, K16.FR.02, K16.FR.04, K16.FR.05, K16.FR.14
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 and EV chooses one of the provided charging schedules.
Prerequisite(s)	Charging station supports ISO15118-20.

Before (Preparations)
Configuration State: N/A
Memory State: N/a
Reusable State(s): State is <i>EVConnectedPreSession</i> State is <i>Authorized15118</i> State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS

Main (Test scenario)	
2. The Charging Station responds with a SetChargingProfileResponse	<p>1. The Test System sends a SetChargingProfileRequest with chargingProfile.chargingProfilePurpose TxProfile chargingProfile.transactionId <Provided transactionId from before> chargingProfile.chargingSchedule[0].id 1 chargingProfile.chargingSchedule[0].chargingRateUnit <Configured chargingRateUnit> chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].startPeriod 0 chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].limit 8 * <limit multiplier> Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></p> <p>chargingProfile.chargingSchedule[1].id 2 chargingProfile.chargingSchedule[1].chargingRateUnit <Configured chargingRateUnit> chargingProfile.chargingSchedule[1].chargingSchedulePeriod[0].startPeriod 0 chargingProfile.chargingSchedule[1].chargingSchedulePeriod[0].limit 7 * <limit multiplier> Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></p> <p>chargingProfile.chargingSchedule[2].id 3 chargingProfile.chargingSchedule[2].chargingRateUnit <Configured chargingRateUnit> chargingProfile.chargingSchedule[2].chargingSchedulePeriod[0].startPeriod 0 chargingProfile.chargingSchedule[2].chargingSchedulePeriod[0].limit 6 * <limit multiplier> Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></p>
Manual Action: Use EV to select charging schedule 2 and calculate a charging schedule within bounds of the selected charging schedule	
3. The Charging Station sends a NotifyEVChargingScheduleRequest .	4. The Test System responds with a NotifyEVChargingScheduleResponse . With status Accepted

Tool validations
<p>* Step 2: Message: SetChargingProfileResponse - status must be Accepted The EV must indicate that the Charging Station has initiated charging schedule renegotiation.</p> <p>* Step 3: Message: NotifyEVChargingScheduleRequest - evseld must be <Configured evseld> - selectedChargingScheduleId SHOULD be 2 - chargingSchedule must fit in <chargingSchedule[2]> <u>Note:</u> K16.FR.04 for selectedChargingProfile is a SHOULD requirement and only triggers a warning</p>
<p>Post scenario validations: N/a</p>

TC_K_114_CS: Renegotiating a Charging Schedule ISO 15118-20 - Initiated by EV

Test case name	Renegotiating a Charging Schedule ISO 15118-20 - Initiated by EV
Test case Id	TC_K_114_CS
Use case Id(s)	K17
Requirement(s)	K17.FR.01, K17.FR.06, K17.FR.09, K17.FR.10, K17.FR.18
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)	Charging station supports ISO15118-20.

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>EVConnectedPreSession</i> State is <i>Authorized15118</i> (for 15118-20) State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action:</u> Use EV to trigger renegotiation requesting: controlMode ScheduledControl	
1. The Charging Station sends a NotifyEVChargingNeedsRequest .	2. The Test System responds with a NotifyEVChargingNeedsResponse . With status Accepted

Main (Test scenario)	
4. The Charging Station responds with a SetChargingProfileResponse	<p>3. The Test System sends a SetChargingProfileRequest with chargingProfile.chargingProfilePurpose TxProfile chargingProfile.transactionId <Provided transactionId from before> chargingProfile.chargingSchedule[0].id 10 chargingProfile.chargingSchedule[0].chargingRateUnit <Configured chargingRateUnit> chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].startPeriod 0 chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].limit 8 * <limit multiplier> Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier> chargingProfile.chargingSchedule[1].id 11 chargingProfile.chargingSchedule[1].chargingRateUnit <Configured chargingRateUnit> chargingProfile.chargingSchedule[1].chargingSchedulePeriod[0].startPeriod 0 chargingProfile.chargingSchedule[1].chargingSchedulePeriod[0].limit 7 * <limit multiplier> Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier> chargingProfile.chargingSchedule[2].id 12 chargingProfile.chargingSchedule[2].chargingRateUnit <Configured chargingRateUnit> chargingProfile.chargingSchedule[2].chargingSchedulePeriod[0].startPeriod 0 chargingProfile.chargingSchedule[2].chargingSchedulePeriod[0].limit 6 * <limit multiplier> Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></p>
<u>Manual Action:</u> Use EV to select charging schedule 11 and calculate a charging schedule within bounds of the selected charging schedule	
5. The Charging Station sends a NotifyEVChargingScheduleRequest .	6. The Test System responds with a NotifyEVChargingScheduleResponse . With status Accepted

Tool validations
<p>* Step 1: Message: NotifyEVChargingNeedsRequest - evseld must be <Configured evseld> - chargingNeeds.acChargingParameters must be <omitted> - chargingNeeds.dcChargingParameters must be <omitted> - chargingNeeds.v2xChargingParameters must be <not omitted> - chargingNeeds.controlMode must be <i>ScheduledControl</i></p> <p>* Step 4: Message: SetChargingProfileResponse - status must be <i>Accepted</i></p> <p>* Step 5: Message: NotifyEVChargingScheduleRequest - evseld must be <Configured evseld> - selectedChargingScheduleId must be 11 or be absent</p>
<p>Post scenario validations: N/a</p>

TC_K_115_CS: ISO 15118-20 Dynamic Control Mode - Success

Test case name	ISO 15118-20 Dynamic Control Mode - Success
Test case Id	TC_K_115_CS
Use case Id(s)	K17,K19
Requirement(s)	K17.FR.01, K17.FR.06, K17.FR.09, K17.FR.10, K17.FR.18, K19.FR.01, K19.FR.06, K19.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 support DynamicControl when the EV is requesting it.
Prerequisite(s)	Charging station supports ISO15118-20.

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>EVConnectedPreSession</i> State is <i>Authorized15118</i> (for 15118-20) State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action:</u> Use EV to trigger renegotiation requesting: <i>controlMode DynamicControl</i>	
1. The Charging Station sends a NotifyEVChargingNeedsRequest .	2. The Test System responds with a NotifyEVChargingNeedsResponse . With status Accepted
4. The Charging Station responds with a SetChargingProfileResponse	3. The Test System sends a SetChargingProfileRequest with chargingProfile.chargingProfilePurpose <i>TxProfile</i> chargingProfile.transactionId <i><Provided transactionId from before></i> chargingProfile.chargingSchedule[0].id <i>10</i> chargingProfile.chargingSchedule[0].chargingRateUnit <i><Configured chargingRateUnit></i> chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].startPeriod <i>0</i> chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].limit <i>8 * <limit multiplier></i> <i>Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></i>
<u>Note:</u> EV to selects charging schedule 10	
5. The Charging Station sends a NotifyEVChargingScheduleRequest .	6. The Test System responds with a NotifyEVChargingScheduleResponse . With status Accepted

Tool validations
<p>* Step 1:</p> <p>Message: NotifyEVChargingNeedsRequest</p> <ul style="list-style-type: none">- evseld must be <i><Configured evseld></i>- chargingNeeds.acChargingParameters must be <i><omitted></i>- chargingNeeds.dcChargingParameters must be <i><omitted></i>- chargingNeeds.v2xChargingParameters must be <i><not omitted></i>- chargingNeeds.controlMode must be <i>DynamicControl</i> <p>* Step 4:</p> <p>Message: SetChargingProfileResponse</p> <ul style="list-style-type: none">- status must be <i>Accepted</i> <p>* Step 5:</p> <p>Message: NotifyEVChargingScheduleRequest</p> <ul style="list-style-type: none">- evseld must be <i><Configured evseld></i>- selectedChargingScheduleId must be <i>10</i>- chargingSchedule must exactly match chargingSchedule[0] from step 4
<p>Post scenario validations:</p> <p>N/a</p>

TC_K_116_CS: Renegotiating a Charging Schedule ISO 15118-20 - Adjusting charging schedule when energy needs change

Test case name	Renegotiating a Charging Schedule ISO 15118-20 - Adjusting charging schedule when energy needs change
Test case Id	TC_K_116_CS
Use case Id(s)	K17,K20
Requirement(s)	K20.FR.01, K20.FR.02, 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)	Charging station supports ISO15118-20.

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>EVConnectedPreSession</i> State is <i>Authorized15118</i> (for 15118-20) State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action:</u> Use EV to trigger renegotiation requesting: <i>departureTime <Current DateTime + 2 hours></i> <i>evTargetEnergyRequest* 5000</i>	
1. The Charging Station sends a NotifyEVChargingNeedsRequest .	2. The Test System responds with a NotifyEVChargingNeedsResponse . With status <i>Accepted</i>
4. The Charging Station responds with a SetChargingProfileResponse	3. The Test System sends a SetChargingProfileRequest with chargingProfile.chargingProfilePurpose <i>TxProfile</i> chargingProfile.transactionId <i><Provided transactionId from before></i> chargingProfile.chargingSchedule[0].id <i>10</i> chargingProfile.chargingSchedule[0].chargingRateUnit <i><Configured chargingRateUnit></i> chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].startPeriod <i>0</i> chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].limit <i>2.5 * <limit multiplier></i> <i>Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></i>
<u>Note:</u> EV selects charging schedule 10	
5. The Charging Station sends a NotifyEVChargingScheduleRequest .	6. The Test System responds with a NotifyEVChargingScheduleResponse . With status <i>Accepted</i>

Main (Test scenario)	
<u>Manual Action:</u> Use EV to trigger renegotiation requesting: departureTime <Current DateTime + 2 hours> evTargetEnergyRequest 10000	
7. The Charging Station sends a NotifyEVChargingNeedsRequest .	8. The Test System responds with a NotifyEVChargingNeedsResponse . With status Accepted
10. The Charging Station responds with a SetChargingProfileResponse	9. The Test System sends a SetChargingProfileRequest with chargingProfile.chargingProfilePurpose TxProfile chargingProfile.transactionId <Provided transactionId from before> chargingProfile.chargingSchedule[0].id 10 chargingProfile.chargingSchedule[0].chargingRateUnit <Configured chargingRateUnit> chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].startPeriod 0 chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].limit 5 * <limit multiplier> Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier>
<u>Note:</u> EV selects charging schedule 10	
11. The Charging Station sends a NotifyEVChargingScheduleRequest .	12. The Test System responds with a NotifyEVChargingScheduleResponse . With status Accepted
<u>Manual Action:</u> Use EV to trigger renegotiation requesting: departureTime <Current DateTime + 5 hours> evTargetEnergyRequest 10000	
13. The Charging Station sends a NotifyEVChargingNeedsRequest .	14. The Test System responds with a NotifyEVChargingNeedsResponse . With status Accepted
16. The Charging Station responds with a SetChargingProfileResponse	15. The Test System sends a SetChargingProfileRequest with chargingProfile.chargingProfilePurpose TxProfile chargingProfile.transactionId <Provided transactionId from before> chargingProfile.chargingSchedule[0].id 10 chargingProfile.chargingSchedule[0].chargingRateUnit <Configured chargingRateUnit> chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].startPeriod 0 chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].limit 2 * <limit multiplier> Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier>
<u>Note:</u> EV selects charging schedule 10	
17. The Charging Station sends a NotifyEVChargingScheduleRequest .	18. The Test System responds with a NotifyEVChargingScheduleResponse . With status Accepted

Tool validations

* Step 1:

Message: **NotifyEVChargingNeedsRequest**

- **evseld** must be *<Configured evseld>*
- **chargingNeeds.acChargingParameters** must be *<omitted>*
- **chargingNeeds.dcChargingParameters** must be *<omitted>*
- **chargingNeeds.v2xChargingParameters.evTargetEnergyRequest** must be *5000*
- **chargingNeeds.controlMode** must be *ScheduledControl*
- **chargingNeeds.departureTime** must be *<Current DateTime + 2 hours>*

* Step 4:

Message: **SetChargingProfileResponse**

- **status** must be *Accepted*

* Step 5:

Message: **NotifyEVChargingScheduleRequest**

- **evseld** must be *<Configured evseld>*
- **chargingSchedule** must be *<not omitted>*

* Step 7:

Message: **NotifyEVChargingNeedsRequest**

- **evseld** must be *<Configured evseld>*
- **chargingNeeds.acChargingParameters** must be *<omitted>*
- **chargingNeeds.dcChargingParameters** must be *<omitted>*
- **chargingNeeds.v2xChargingParameters.evTargetEnergyRequest** must be *10000*
- **chargingNeeds.controlMode** must be *ScheduledControl*
- **chargingNeeds.departureTime** must be *<Current DateTime + 2 hours>*

* Step 10:

Message: **SetChargingProfileResponse**

- **status** must be *Accepted*

* Step 11:

Message: **NotifyEVChargingScheduleRequest**

- **evseld** must be *<Configured evseld>*
- **chargingSchedule** must be *<not omitted>*

* Step 13:

Message: **NotifyEVChargingNeedsRequest**

- **evseld** must be *<Configured evseld>*
- **chargingNeeds.acChargingParameters** must be *<omitted>*
- **chargingNeeds.dcChargingParameters** must be *<omitted>*
- **chargingNeeds.v2xChargingParameters.evTargetEnergyRequest** must be *10000*
- **chargingNeeds.controlMode** must be *ScheduledControl*
- **chargingNeeds.departureTime** must be *<Current DateTime + 5 hours>*

* Step 16:

Message: **SetChargingProfileResponse**

- **status** must be *Accepted*

* Step 17:

Message: **NotifyEVChargingScheduleRequest**

- **evseld** must be *<Configured evseld>*
- **chargingSchedule** must be *<not omitted>*

Post scenario validations:

N/a

TC_K_118_CS: Priority Charging - Requesting priority charging remotely

Test case name	Priority Charging - Requesting priority charging remotely
Test case Id	TC_K_118_CS
Use case Id(s)	K21
Requirement(s)	K21.FR.01, K21.FR.02, K21.FR.03, K21.FR.04, K21.FR.05, K21.FR.06, K21.FR.07, K21.FR.08, K21.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 supports priority charging initiated from CSMS.
Prerequisite(s)	n/a

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

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a UsePriorityChargingResponse	1. The Test System sends a UsePriorityChargingRequest with transactionId <i><unknown transactionId></i> activate <i>true</i>
4. The Charging Station responds with a UsePriorityChargingResponse	3. The Test System sends a UsePriorityChargingRequest with transactionId <i><transactionId from preparation></i> activate <i>true</i>
6. The Charging Station responds with a SetChargingProfileResponse	5. The Test System sends a SetChargingProfileRequest with evseld <i>0</i> chargingProfile.chargingProfilePurpose <i>PriorityCharging</i> chargingProfile.chargingSchedule.chargingRateUnit <i><Configured chargingRateUnit></i> chargingProfile.chargingSchedule.chargingSchedulePeriod.startPeriod <i>0</i> chargingProfile.chargingSchedule.chargingSchedulePeriod.limit <i>6 * <limit multiplier></i> Note: Check Determine Charging Profile Limit Multiplier for <i><limit multiplier></i>
8. The Charging Station responds with a UsePriorityChargingResponse	7. The Test System sends a UsePriorityChargingRequest with transactionId <i><transactionId from preparation></i> activate <i>true</i>
9. The Charging Station sends a NotifyPriorityChargingRequest	10. The Test System responds with a NotifyPriorityChargingResponse
12. The Charging Station responds with a UsePriorityChargingResponse	11. The Test System sends a UsePriorityChargingRequest with transactionId <i><transactionId from preparation></i> activate <i>false</i>
13. The Charging Station sends a NotifyPriorityChargingRequest	14. The Test System responds with a NotifyPriorityChargingResponse

Tool validations
<div><div>* Step 2:</div><div>Message: UsePriorityChargingResponse</div><div>status must be <i>Rejected</i></div><div>statusInfo.reasonCode must be <i>TxNotFound</i></div><div>* Step 4:</div><div>Message: UsePriorityChargingResponse</div><div>status must be <i>NoProfile</i></div><div>statusInfo.reasonCode must be <i>NotFound</i></div><div>* Step 6:</div><div>Message: SetChargingProfileResponse</div><div>status must be <i>Accepted</i></div><div>* Step 8:</div><div>Message: UsePriorityChargingResponse</div><div>status must be <i>Accepted</i></div><div>* Step 9:</div><div>Message: NotifyPriorityChargingRequest</div><div>transactionId must be <i><transactionId from preparation></i></div><div>activated must be <i>true</i></div><div>* Step 12:</div><div>Message: UsePriorityChargingResponse</div><div>status must be <i>Accepted</i></div><div>* Step 13:</div><div>Message: NotifyPriorityChargingRequest</div><div>transactionId must be <i><transactionId from preparation></i></div><div>activated must be <i>false</i></div></div>
<div><div>Post scenario validations:</div><div>N/a</div></div>

TC_K_119_CS: Priority Charging - Requesting priority charging locally

Test case name	Priority Charging - Requesting priority charging locally
Test case Id	TC_K_119_CS
Use case Id(s)	K22
Requirement(s)	K22.FR.01, K22.FR.02, K22.FR.03, K22.FR.04, K22.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 supports priority charging initiated locally.
Prerequisite(s)	n/a

Before (Preparations)
Configuration State:
Memory State: A chargingprofile with: ChargingProfilePurpose TxDefaultProfile AND StackLevel 1 AND Duration <omitted> AND limit 6
Reusable State(s): State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action</u> : Request the CS to start priority charging for the transaction	
<u>Note(s)</u> : - The Charging Station will notify the EV driver that it is not capable of priority charging at this moment.	
2. The Charging Station responds with a SetChargingProfileResponse	1. The Test System sends a SetChargingProfileRequest with evseld 0 chargingProfile.chargingProfilePurpose <i>PriorityCharging</i> chargingProfile.chargingSchedule.chargingRateUnit <i><Configured chargingRateUnit></i> chargingProfile.chargingSchedule.duration <i><Omitted></i> chargingProfile.chargingSchedule.chargingSchedulePeriod.startPeriod 0 chargingProfile.chargingSchedule.chargingSchedulePeriod.limitPeriod 10 * <limit multiplier> <i>Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></i>
<u>Manual Action</u> : Request the CS to start priority charging for the transaction	
3. The Charging Station sends a NotifyPriorityChargingRequest	4. The Test System responds with a NotifyPriorityChargingResponse
6. The Charging Station responds with a GetCompositeScheduleResponse	5. The Test System sends a GetCompositeScheduleRequest with evseld <i><Configured evseld></i> duration 60 chargingRateUnit <i><Configured chargingRateUnit></i>
<u>Manual Action</u> : Request the CS to stop priority charging for the transaction	

Main (Test scenario)	
7. The Charging Station sends a NotifyPriorityChargingRequest	8. The Test System responds with a NotifyPriorityChargingResponse
10. The Charging Station responds with a GetCompositeScheduleResponse	9. The Test System sends a GetCompositeScheduleRequest with evseld <Configured evseld> duration 60 chargingRateUnit <Configured chargingRateUnit>

Tool validations
<p>* Step 2: Message: SetChargingProfileResponse status must be <i>Accepted</i></p> <p>* Step 3: Message: NotifyPriorityChargingRequest transactionId must be <transactionId> activated must be <i>true</i></p> <p>* Step 6: Message: GetCompositeScheduleResponse status must be <i>Accepted</i> schedule.evseld must be <Configured evseld> schedule.duration must be 60 schedule.chargingRateUnit must be <Configured chargingRateUnit> schedule.chargingSchedulePeriod[0].startPeriod 0 schedule.chargingSchedulePeriod[0].limit 10 * <limit multiplier> Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></p> <p>* Step 7: Message: NotifyPriorityChargingRequest transactionId must be <transactionId> activated must be <i>false</i></p> <p>* Step 10: Message: GetCompositeScheduleResponse status must be <i>Accepted</i> schedule.evseld must be <Configured evseld> schedule.duration must be 60 schedule.chargingRateUnit must be <Configured chargingRateUnit> schedule.chargingSchedulePeriod[0].limit must not be 6 * <limit multiplier> Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></p> <p>Post scenario validations: N/a</p>

TC_K_120_CS: EMS Control - Smart Charging with EMS and LocalGeneration

Test case name	EMS Control - Smart Charging with EMS and LocalGeneration
Test case Id	TC_K_120_CS
Use case Id(s)	K27
Requirement(s)	K27.FR.01, K27.FR.02, K27.FR.03, K27.FR.04, K27.FR.05
System under test	Charging Station
Description	To show how locally available capacity can be taken into account.
Purpose	To verify if the Charging Station is able to inform the CSMS when it's capacity is being throttled by a locally connected EMS.
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station supports connecting locally a EMS. - EMS hasn't set any limit on the charger.

Before (Preparations)
Configuration State: SmartChargingCtrlr.LimitChangeSignificance is 0 SmartChargingCtrlr.NotifyChargingLimitWithSchedules is true
Memory State: N/a
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a GetChargingProfilesResponse	1. The Test System sends a GetChargingProfilesRequest with evseld <Configured evseld> chargingProfile.chargingProfilePurpose LocalGeneration
<u>Manual Action:</u> Use EMS to report LocalGeneration to the charger with limit 6 * <limit multiplier> Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier>	
3. The Charging Station sends a NotifyChargingLimitRequest	4. The Test System responds with a NotifyChargingLimitResponse
6. The Charging Station responds with a GetChargingProfilesResponse	5. The Test System sends a GetChargingProfilesRequest with evseld <Configured evseld> chargingProfile.chargingProfilePurpose LocalGeneration
7. The Charging Station sends a ReportChargingProfilesRequest	8. The Test System responds with a ReportChargingProfilesResponse
<u>Note(s):</u> - If ReportChargingProfilesRequest.tbc is True in Step 7 then step 7 and 8 will be repeated	
<u>Manual Action:</u> Use EMS to report LocalGeneration to the charger with limit 7 * <limit multiplier> and use EMS to report ChargingStation-ExternalConstraints to the charger with limit 16 * <limit multiplier> Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier>	
9. The Charging Station sends a NotifyChargingLimitRequest	10. The Test System responds with a NotifyChargingLimitResponse
11. The Charging Station sends a NotifyChargingLimitRequest	12. The Test System responds with a NotifyChargingLimitResponse

Tool validations

* Step 2:

Message: **GetChargingProfilesResponse**

- **status** must be *NoProfiles*

* Step 3:

Message: **NotifyChargingLimitRequest**

- **evseld** must be *<Configured evseld>*

- **chargingLimit.chargingLimitSource** must be *EMS* or *Other*

- **chargingLimit.isLocalGeneration** must be *true*

There must be one **ChargingSchedulePeriod** be reported with

- **chargingSchedule.chargingSchedulePeriod.limit** must be $6 * \text{<limit multiplier>}$

Note: Check [Determine Charging Profile Limit Multiplier](#) for *<limit multiplier>*_

* Step 6:

Message: **GetChargingProfilesResponse**

- **status** must be *Accepted*

* Step 7:

Message **ReportChargingProfilesRequest**

- **requestId** must be *Same Id as in the GetChargingProfilesRequest in step 3*

- **EVSEId** must be *<Configured evseld>*

There must be one **ChargingProfile** be reported with

- **chargingProfile[0].chargingProfilePurpose** must be *LocalGeneration*

* Step 9:

Message: **NotifyChargingLimitRequest**

- **evseld** must be *<Configured evseld>*

- **chargingLimit.chargingLimitSource** must be *EMS* or *Other*

IF *<NotifyChargingLimitRequest.chargingLimit.isLocalGeneration>* is true

There must be one **ChargingSchedulePeriod** be reported with

- **chargingSchedule.chargingSchedulePeriod.limit** must be $7 * \text{<limit multiplier>}$

Note: Check [Determine Charging Profile Limit Multiplier](#) for *<limit multiplier>*

ELSE

There must be one **ChargingSchedulePeriod** be reported with

- **chargingSchedule.chargingSchedulePeriod.limit** must be $16 * \text{<limit multiplier>}$

Note: Check [Determine Charging Profile Limit Multiplier](#) for *<limit multiplier>*

END IF

* Step 11:

Message: **NotifyChargingLimitRequest**

- **evseld** must be *<Configured evseld>*

- **chargingLimit.chargingLimitSource** must be *EMS* or *Other*

IF *<NotifyChargingLimitRequest.chargingLimit.isLocalGeneration>* is true

There must be one **ChargingSchedulePeriod** be reported with

- **chargingSchedule.chargingSchedulePeriod.limit** must be $7 * \text{<limit multiplier>}$

Note: Check [Determine Charging Profile Limit Multiplier](#) for *<limit multiplier>*

ELSE

There must be one **ChargingSchedulePeriod** be reported with

- **chargingSchedule.chargingSchedulePeriod.limit** must be $16 * \text{<limit multiplier>}$

Note: Check [Determine Charging Profile Limit Multiplier](#) for *<limit multiplier>*

END IF

Post scenario validations:

N/a

TC_K_121_CS: Dynamic charging profiles from CSMS - Pull

Test case name	Dynamic charging profiles from CSMS - Pull
Test case Id	TC_K_121_CS
Use case Id(s)	K28
Requirement(s)	K28.FR.08, K28.FR.09, K28.FR.10
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 support pulling DynamicControl charging profiles from the CSMS.
Prerequisite(s)	SmartChargingCtrlr.SupportsFeature[DynamicProfiles] is 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 SetChargingProfileResponse	<p>1. The Test System sends a SetChargingProfileRequest with</p> <p>chargingProfile.id 123</p> <p>chargingProfile.chargingProfilePurpose <i>ChargingStationMaxProfile</i></p> <p>chargingProfile.chargingProfileKind <i>Dynamic</i></p> <p>chargingProfile.dynUpdateTime <i><Current dateTime></i></p> <p>chargingProfile.dynUpdateInterval 30</p> <p>chargingProfile.chargingSchedule[0].chargingRateUnit <i><Configured chargingRateUnit></i></p> <p>chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].startPeriod 0</p> <p>chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].limit 8 * <i><limit multiplier></i></p> <p>Note: Check Determine Charging Profile Limit Multiplier for <i><limit multiplier></i></p>
<u>Note(s)</u> : The Test System waits 30 seconds.	
3. The Charging Station sends a PullDynamicScheduleUpdateRequest	<p>4. The Test System responds with a PullDynamicScheduleUpdateResponse with</p> <p>status <i>Accepted</i></p> <p>scheduleUpdate.limit 6 * <i><limit multiplier></i></p> <p>Note: Check Determine Charging Profile Limit Multiplier for <i><limit multiplier></i></p>
6. The Charging Station responds with a GetChargingProfilesResponse	5. The Test System sends a GetChargingProfilesRequest with chargingProfile.chargingProfileId 123 requestId <i><Generated requestId></i>
7. The Charging Station sends a ReportChargingProfilesRequest	8. The Test System responds with a ReportChargingProfilesResponse
<u>Note(s)</u> : - If tbc is True at Step 7 then step 7 and 8 will be repeated	

Tool validations
<p>* Step 2:</p> <p>Message: SetChargingProfileResponse</p> <ul style="list-style-type: none">- status must be <i>Accepted</i> <p>* Step 3:</p> <p>Message: PullDynamicScheduleUpdateRequest</p> <ul style="list-style-type: none">- chargingProfileId must be 123 <p>* Step 6:</p> <p>Message: GetChargingProfilesResponse</p> <ul style="list-style-type: none">- status must be <i>Accepted</i> <p>* Step 7:</p> <p>Message: ReportChargingProfilesRequest</p> <p>Note: The date/time when receiving the PullDynamicScheduleUpdateResponse in step 4 is called x:</p> <ul style="list-style-type: none">- chargingProfile.dynUpdateTime must be <x>- chargingProfile.id must be 123- chargingProfile.chargingSchedule.chargingSchedulePeriod[0].limit must be 6 * <limit multiplier> <p>Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></p>
<p>Post scenario validations:</p> <p>N/a</p>

TC_K_122_CS: Dynamic charging profiles from CSMS - Push

Test case name	Dynamic charging profiles from CSMS - Push
Test case Id	TC_K_122_CS
Use case Id(s)	K28
Requirement(s)	K28.FR.05, K28.FR.06, K28.FR.09, K28.FR.13, K28.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 support DynamicControl charging profiles with the CSMS pushing changes to the CS.
Prerequisite(s)	SmartChargingCtrlr.SupportsFeature[DynamicProfiles] is true

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

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetChargingProfileResponse	<p>1. The Test System sends a SetChargingProfileRequest with</p> <p>chargingProfile.id 122</p> <p>chargingProfile.stackLevel 0</p> <p>chargingProfile.chargingProfilePurpose <i>ChargingStationMaxProfile</i></p> <p>chargingProfile.chargingProfileKind <i>Dynamic</i></p> <p>chargingProfile.chargingSchedule[0].chargingRateUnit <Configured chargingRateUnit></p> <p>chargingProfile.chargingSchedule[0].duration <omitted></p> <p>chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].startPeriod 0</p> <p>chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].limit 6 * <limit multiplier></p> <p>Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></p>
4. The Charging Station responds with a SetChargingProfileResponse	<p>3. The Test System sends a SetChargingProfileRequest with</p> <p>chargingProfile.id 123</p> <p>chargingProfile.stackLevel 1</p> <p>chargingProfile.chargingProfilePurpose <i>ChargingStationMaxProfile</i></p> <p>chargingProfile.chargingProfileKind <i>Dynamic</i></p> <p>chargingProfile.chargingSchedule[0].chargingRateUnit <Configured chargingRateUnit></p> <p>chargingProfile.chargingSchedule[0].duration 15</p> <p>chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].startPeriod 0</p> <p>chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].limit 8 * <limit multiplier></p> <p>Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></p>

Main (Test scenario)	
6. The Charging Station responds with a GetChargingProfilesResponse	5. The Test System sends a GetChargingProfilesRequest with chargingProfile.chargingProfileId 123 requestId <Generated requestId>
7. The Charging Station sends a ReportChargingProfilesRequest	8. The Test System responds with a ReportChargingProfilesResponse
<u>Note(s):</u> - If tbc is True at Step 7 then step 7 and 8 will be repeated	
10 The Charging Station responds with a UpdateDynamicScheduleResponse	9 The Test System sends a UpdateDynamicScheduleRequest with chargingProfileId 123 scheduleUpdate.limit 8 * <limit multiplier> Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier>
12. The Charging Station responds with a GetChargingProfilesResponse	11. The Test System sends a GetChargingProfilesRequest with chargingProfile.chargingProfileId 123 requestId <Generated requestId>
13. The Charging Station sends a ReportChargingProfilesRequest	14. The Test System responds with a ReportChargingProfilesResponse
<u>Note(s):</u> - If tbc is True at Step 13 then step 13 and 14 will be repeated	
<u>Note:</u> The Test System waits 20 seconds	
16. The Charging Station responds with a GetCompositeScheduleResponse	15. The Test System sends a GetCompositeScheduleRequest with evseld <Configured evseld> duration is 900 chargingRateUnit <Configured chargingRateUnit>
18 The Charging Station responds with a UpdateDynamicScheduleResponse	17 The Test System sends a UpdateDynamicScheduleRequest with chargingProfileId 123 scheduleUpdate.limit 8 * <limit multiplier> Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier>
20. The Charging Station responds with a GetCompositeScheduleResponse	19. The Test System sends a GetCompositeScheduleRequest with evseld <Configured evseld> duration is 900 chargingRateUnit <Configured chargingRateUnit>

Tool validations

* Step 2:

Message: **SetChargingProfileResponse**

- **status** must be *Accepted*

* Step 4:

Message: **SetChargingProfileResponse**

- **status** must be *Accepted*

* Step 6:

Message: **GetChargingProfilesResponse**

- **status** must be *Accepted*

* Step 7:

Note: The date/time when sending the **SetChargingProfileRequest** in step 1 is called **x**:

Message: **ReportChargingProfilesRequest**

- **chargingProfile.dynUpdateTime** must be <x>

* Step 10:

Message: **UpdateDynamicScheduleResponse**

- **status** must be *Accepted*

* Step 12:

Message: **GetChargingProfilesResponse**

- **status** must be *Accepted*

* Step 13:

Note: The date/time when sending the **UpdateDynamicScheduleRequest** in step 7 is called **x**:

Message: **ReportChargingProfilesRequest**

- **chargingProfile.dynUpdateTime** must be <x>

* Step 16:

Message: **GetCompositeScheduleResponse**

- **status** must be *Accepted*

- **schedule.chargingRateUnit** must be <Configured chargingRateUnit>

- **schedule.chargingSchedulePeriod[0].startPeriod** must be 0

- **schedule.chargingSchedulePeriod[0].limit** must be $6 * \text{<limit multiplier>}$

Note: Check [Determine Charging Profile Limit Multiplier](#) for <limit multiplier>

* Step 18:

Message: **UpdateDynamicScheduleResponse**

- **status** must be *Accepted*

* Step 20:

Message: **GetCompositeScheduleResponse**

- **status** must be *Accepted*

- **schedule.chargingRateUnit** must be <Configured chargingRateUnit>

- **schedule.chargingSchedulePeriod[0].startPeriod** must be 0

- **schedule.chargingSchedulePeriod[0].limit** must be $8 * \text{<limit multiplier>}$

Note: Check [Determine Charging Profile Limit Multiplier](#) for <limit multiplier>

Post scenario validations:

N/a

TC_K_123_CS: Dynamic charging profiles from CSMS - validations

Test case name	Dynamic charging profiles from CSMS - validations
Test case Id	TC_K_123_CS
Use case Id(s)	K28
Requirement(s)	K28.FR.01, K28.FR.02, K28.FR.03, K28.FR.04, K28.FR.11
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 support DynamicControl charging profiles.
Prerequisite(s)	SmartChargingCtrlr.SupportsFeature[DynamicProfiles] is true

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

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetChargingProfileResponse	<p>1. The Test System sends a SetChargingProfileRequest with chargingProfile.chargingProfilePurpose <i>ChargingStationMaxProfile</i></p> <p>chargingProfile.chargingProfileKind <i>Dynamic</i></p> <p>chargingProfile.chargingSchedule[0].chargingRateUnit <i><Configured chargingRateUnit></i></p> <p>chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].startPeriod <i>0</i></p> <p>chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].limit <i>8 * <limit multiplier></i></p> <p>Note: Check Determine Charging Profile Limit Multiplier for <i><limit multiplier></i></p> <p>chargingProfile.chargingSchedule[0].chargingSchedulePeriod[1].startPeriod <i>900</i></p> <p>chargingProfile.chargingSchedule[0].chargingSchedulePeriod[1].limit <i>4 * <limit multiplier></i></p> <p>Note: Check Determine Charging Profile Limit Multiplier for <i><limit multiplier></i></p>

Main (Test scenario)	
4. The Charging Station responds with a SetChargingProfileResponse	<p>3. The Test System sends a SetChargingProfileRequest with chargingProfile.chargingProfilePurpose <i>ChargingStationMaxProfile</i> chargingProfile.chargingProfileKind <i>Dynamic</i> chargingProfile.chargingSchedule[0].chargingRateUnit <i><Configured chargingRateUnit></i> chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].startPeriod <i>0</i> chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].limit <i>8 * <limit multiplier></i> <i>Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></i></p> <p>chargingProfile.chargingSchedule[1].chargingRateUnit <i><Configured chargingRateUnit></i> chargingProfile.chargingSchedule[1].chargingSchedulePeriod[0].startPeriod <i>0</i> chargingProfile.chargingSchedule[1].chargingSchedulePeriod[0].limit <i>16 * <limit multiplier></i> <i>Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></i></p>
6. The Charging Station responds with a SetChargingProfileResponse	<p>5. The Test System sends a SetChargingProfileRequest with chargingProfile.chargingProfilePurpose <i>ChargingStationMaxProfile</i> chargingProfile.chargingProfileKind <i>Dynamic</i> chargingProfile.chargingSchedule[0].chargingRateUnit <i><Configured chargingRateUnit></i> chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].startPeriod <i>1</i> chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].limit <i>8 * <limit multiplier></i> <i>Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></i></p>
8. The Charging Station responds with a SetChargingProfileResponse	<p>7. The Test System sends a SetChargingProfileRequest with chargingProfile.chargingProfilePurpose <i>ChargingStationMaxProfile</i> chargingProfile.chargingProfileKind <i>Relative</i> chargingProfile.dynUpdateInterval <i>10</i> chargingProfile.chargingSchedule.chargingRateUnit <i><Configured chargingRateUnit></i> chargingProfile.chargingSchedule.chargingSchedulePeriod.startPeriod <i>0</i> chargingProfile.chargingSchedule.chargingSchedulePeriod.limit <i>6 * <limit multiplier></i> <i>Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></i></p>

Main (Test scenario)	
10. The Charging Station responds with a SetChargingProfileResponse	<p>9. The Test System sends a SetChargingProfileRequest with</p> <p>chargingProfile.id 123</p> <p>chargingProfile.chargingProfilePurpose <i>ChargingStationMaxProfile</i></p> <p>chargingProfile.chargingProfileKind <i>Relative</i></p> <p>chargingProfile.chargingSchedule.chargingRateUnit <i><Configured chargingRateUnit></i></p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod.startPeriod 0</p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod.limit 6 * <i><limit multiplier></i></p> <p>Note: Check Determine Charging Profile Limit Multiplier for <i><limit multiplier></i></p>
12 The Charging Station responds with a UpdateDynamicScheduleResponse	<p>11 The Test System sends a UpdateDynamicScheduleRequest with</p> <p>chargingProfileId 123</p> <p>scheduleUpdate.limit 8 * <i><limit multiplier></i></p> <p>Note: Check Determine Charging Profile Limit Multiplier for <i><limit multiplier></i></p>

Tool validations
<p>* Step 2: (2 periods)</p> <p>Message: SetChargingProfileResponse</p> <p>- status must be <i>Rejected</i></p> <p>* Step 4: (2 schedules)</p> <p>Message: SetChargingProfileResponse</p> <p>- status must be <i>Rejected</i></p> <p>* Step 6: (startPeriod not 0)</p> <p>Message: SetChargingProfileResponse</p> <p>- status must be <i>Rejected</i></p> <p>* Step 8: (dynUpdateInterval in Relative)</p> <p>Message: SetChargingProfileResponse</p> <p>- status must be <i>Rejected</i></p> <p>* Step 10: (not Dynamic)</p> <p>Message: SetChargingProfileResponse</p> <p>- status must be <i>Accepted</i></p> <p>* Step 12: (no Dynamic profile installed)</p> <p>Message: UpdateDynamicScheduleResponse</p> <p>- status must be <i>Rejected</i></p> <p>Post scenario validations:</p> <p>N/a</p>

TC_K_124_CS: Dynamic charging profiles by external system - No Dynamic charging profile configured

Test case name	Dynamic charging profiles by external system - No Dynamic charging profile configured
Test case Id	TC_K_124_CS
Use case Id(s)	K29
Requirement(s)	K29.FR.01, K29.FR.02, K29.FR.04, K29.FR.05, K29.FR.06, K11.FR.06
System under test	Charging Station
Description	To allow a dynamically changing limit in a charging profile from an external system.
Purpose	To verify if the Charging Station is able to report correct updates or charging profile limits when being throttled by a locally connected EMS.
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station is connected locally to a system which can apply charging limit constraints. - External connected system hasn't set any limit on the charger.

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

Main (Test scenario)	
Charging Station	CSMS
1. <u>Manual Action</u> : Use external connected system to send ChargingStation-ExternalConstraints limit to the charger: 6 * <limit multiplier>.	
3. The Charging Station responds with a GetChargingProfilesResponse	2. The Test System sends a GetChargingProfilesRequest with evseld <Configured evseld> chargingProfile.chargingProfilePurpose ChargingStationExternalConstraints
4. The Charging Station sends a ReportChargingProfilesRequest	5. The Test System responds with a ReportChargingProfilesResponse
<u>Note(s)</u> : - If ReportChargingProfilesRequest.tbc is True in Step 4 then step 4 and 5 will be repeated	
6. <u>Manual Action</u> : Use external connected system to send different ChargingStation-ExternalConstraints limit to the charger: 7 * <limit multiplier>.	
8. The Charging Station responds with a GetChargingProfilesResponse	7. The Test System sends a GetChargingProfilesRequest with evseld <Configured evseld> chargingProfile.chargingProfilePurpose ChargingStationExternalConstraints
9. The Charging Station sends a ReportChargingProfilesRequest	10. The Test System responds with a ReportChargingProfilesResponse
<u>Note(s)</u> : - If ReportChargingProfilesRequest.tbc is True in Step 9 then step 9 and 10 will be repeated	

Tool validations

* Step 3:

Message: **GetChargingProfilesResponse**

- **status** must be *Accepted*

* Step 4:

Message **ReportChargingProfilesRequest**

- **requestId** must be *Same Id as in the GetChargingProfilesRequest in step 2*

- **EVSEId** must be *<Configured evseld>*

There must be one ChargingProfile be reported with

- **chargingProfile[0].chargingProfilePurpose** must be *ChargingStationExternalConstraints*

- **chargingProfile[0].chargingProfileKind** must be *Dynamic* or *Absolute*

IF **chargingProfile[0].chargingProfileKind** is *Dynamic* THEN

- **chargingProfile[0].chargingSchedule** size must be *1*

- **chargingProfile[0].chargingSchedule[0].chargingSchedulePeriod** size must be *1*

- **chargingProfile[0].chargingSchedule[0].chargingSchedulePeriod[0].startPeriod** must be *0*

- **chargingProfile[0].dynUpdateTime** must be *<Datetime of moment of step 1>*

END IF

- **chargingProfile[0].chargingSchedule[0].chargingSchedulePeriod[0].limit** must be *6 * <limit multiplier>*

* Step 8:

Message: **GetChargingProfilesResponse**

- **status** must be *Accepted*

* Step 9:

Message **ReportChargingProfilesRequest**

- **requestId** must be *Same Id as in the GetChargingProfilesRequest in step 7*

- **EVSEId** must be *<Configured evseld>*

There must be one ChargingProfile be reported with

- **chargingProfile[0].chargingProfilePurpose** must be *ChargingStationExternalConstraints*

- **chargingProfile[0].chargingProfileKind** must be *Dynamic* or *Absolute*

IF **chargingProfile[0].chargingProfileKind** is *Dynamic* THEN

- **chargingProfile[0].id** must be *<ReportChargingProfilesRequest.chargingProfile[0].id of step 4>*

- **chargingProfile[0].chargingSchedule** size must be *1*

- **chargingProfile[0].chargingSchedule[0].chargingSchedulePeriod** size must be *1*

- **chargingProfile[0].chargingSchedule[0].chargingSchedulePeriod[0].startPeriod** must be *0*

- **chargingProfile[0].dynUpdateTime** must be *<Datetime of moment of step 6>*

END IF

IF **chargingProfile[0].chargingProfileKind** is *Absolute* THEN

- **chargingProfile[0].id** must be not *<ReportChargingProfilesRequest.chargingProfile[0].id of step 4>*

END IF

- **chargingProfile[0].chargingSchedule[0].chargingSchedulePeriod[0].limit** must be *7 * <limit multiplier>*

Post scenario validations:

N/a

TC_K_125_CS: Dynamic charging profiles by external system - Dynamic charging profile configured

Test case name	Dynamic charging profiles by external system - Dynamic charging profile configured
Test case Id	TC_K_125_CS
Use case Id(s)	K29
Requirement(s)	K29.FR.01, K29.FR.02, K29.FR.03, K29.FR.05, K29.FR.06
System under test	Charging Station
Description	To allow a dynamically changing limit in a charging profile from an external system.
Purpose	To verify if the Charging Station is able to inform the CSMS when it's capacity is being throttled by a locally connected EMS.
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station is connected locally to a system which can apply charging limit constraints. - External connected system hasn't set any limit on the charger.

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 Test System sends a SetChargingProfileRequest with evseld 0 chargingProfile.id <Configured chargingProfileId> chargingProfile.stackLevel 0 chargingProfile.chargingProfilePurpose <i>ChargingStationMaxProfile</i> chargingProfile.chargingProfileKind <i>Dynamic</i> chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].startPeriod 0 chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].operationMode <i>ExternalSetpoint</i>
4. The Charging Station responds with a SetChargingProfileResponse	3. The Test System sends a SetChargingProfileRequest with evseld 0 chargingProfile.id <Configured chargingProfileId2> chargingProfile.stackLevel 1 chargingProfile.chargingProfilePurpose <i>ChargingStationMaxProfile</i> chargingProfile.chargingProfileKind <i>Dynamic</i> chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].startPeriod 0 chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].operationMode <i>ExternalLimits</i>
<u>Manual Action:</u> Use external connected system to send ChargingStation-ExternalConstraints limit to the charger: $8 * \text{<limit multiplier>}$	
<u>Manual Action:</u> Use external connected system to send ChargingStation-ExternalConstraints setPoint to the charger: $10 * \text{<limit multiplier>}$	
6. The Charging Station responds with a GetChargingProfilesResponse	5. The Test System sends a GetChargingProfilesRequest with evseld <Configured evseld> chargingProfile.chargingProfilePurpose <i>ChargingStationMaxProfile</i>

Main (Test scenario)	
7. The Charging Station sends a ReportChargingProfilesRequest	8. The Test System responds with a ReportChargingProfilesResponse
<u>Note(s):</u> - If ReportChargingProfilesRequest.tbc is True in Step 7 then step 7 and 8 will be repeated	

Tool validations
<p>* Step 2: Message: SetChargingProfileResponse - status must be <i>Accepted</i></p> <p>* Step 6: Message: GetChargingProfilesResponse - status must be <i>Accepted</i></p> <p>* Step 7: Message ReportChargingProfilesRequest - requestId must be <i>Same Id as in the GetChargingProfilesRequest in step 5</i> - EVSEId must be <i>0</i> There must be one ChargingProfile be reported with - chargingProfile[0].id must be <i><Configured chargingProfileId></i> - chargingProfile[0].stackLevel must be <i>0</i> - chargingProfile[0].chargingProfilePurpose must be <i>ChargingStationMaxProfile</i> - chargingProfile[0].chargingProfileKind must be <i>Dynamic</i> - chargingProfile[0].dynUpdateTime must be <i><Datetime of moment of step 3></i> - chargingProfile[0].chargingSchedule size must be <i>1</i> - chargingProfile[0].chargingSchedule[0].chargingSchedulePeriod size must be <i>1</i> - chargingProfile[0].chargingSchedule[0].chargingSchedulePeriod[0].startPeriod must be <i>0</i> - chargingProfile[0].chargingSchedule[0].chargingSchedulePeriod[0].operationMode <i>ExternalSetpoint</i> - chargingProfile[0].chargingSchedule[0].chargingSchedulePeriod[0].limit must be <i><omitted></i> - chargingProfile[0].chargingSchedule[0].chargingSchedulePeriod[0].setpoint must be <i>10 * <limit multiplier></i> There must be one ChargingProfile be reported with - chargingProfile[1].id must be <i><Configured chargingProfileId2></i> - chargingProfile[1].stackLevel must be <i>1</i> - chargingProfile[1].chargingProfilePurpose must be <i>ChargingStationMaxProfile</i> - chargingProfile[1].chargingProfileKind must be <i>Dynamic</i> - chargingProfile[1].dynUpdateTime must be <i><Datetime of moment of step 3></i> - chargingProfile[1].chargingSchedule size must be <i>1</i> - chargingProfile[1].chargingSchedule[0].chargingSchedulePeriod size must be <i>1</i> - chargingProfile[1].chargingSchedule[0].chargingSchedulePeriod[0].startPeriod must be <i>0</i> - chargingProfile[1].chargingSchedule[0].chargingSchedulePeriod[0].operationMode <i>ExternalLimits</i> - chargingProfile[1].chargingSchedule[0].chargingSchedulePeriod[0].limit must be <i>8 * <limit multiplier></i> - chargingProfile[1].chargingSchedule[0].chargingSchedulePeriod[0].setpoint must be <i><omitted></i></p> <p>Post scenario validations: N/a</p>

TC_K_129_CS: Set Charging Profile - PriorityCharging persistent over reboot

Test case name	Set Charging Profile - PriorityCharging persistent over reboot
Test case Id	TC_K_129_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 of purpose <code>PriorityCharging</code> persistent over reboot as described at the OCPP specification.
Prerequisite(s)	<code>SmartChargingCtrlr.SupportedAdditionalPurposes</code> contains <code>PriorityCharging</code>

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetChargingProfileResponse	<p>1. The Test System sends a SetChargingProfileRequest with <code>evseld 0</code> <code>chargingProfile.id</code> <Configured chargingProfileId> <code>chargingProfile.chargingProfilePurpose</code> <code>PriorityCharging</code> <code>chargingProfile.chargingSchedule.duration</code> <Configured duration> <code>chargingProfile.chargingSchedule.chargingRateUnit</code> <Configured chargingRateUnit> <code>chargingProfile.chargingSchedule.chargingSchedulePeriod.startPeriod 0</code> <code>chargingProfile.chargingSchedule.chargingSchedulePeriod.limit 6 * <limit multiplier></code> <i>Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></i> <code>chargingProfile.chargingSchedule.chargingSchedulePeriod.numberPhases</code> <Configured numberPhases></p>
3. Execute Reusable State <i>Booted</i>	
5. The Charging Station responds with a GetChargingProfilesResponse	<p>4. The Test System sends a GetChargingProfilesRequest with <code>chargingProfile.chargingProfileId[0]</code> <Configured chargingProfileId> <code>chargingProfile.chargingProfileId[1]</code> <Configured chargingProfileId2></p>
6. The Charging Station sends a ReportChargingProfilesRequest	<p>7. The Test System responds with a ReportChargingProfilesResponse</p>
<p>Note(s): - If <code>tbc</code> is True at Step 6 then step 6 and 7 will be repeated</p>	

Tool validations
<p>* Step 2:</p> <p>Message SetChargingProfileResponse</p> <ul style="list-style-type: none">- status must be <i>Accepted</i> <p>* Step 5:</p> <p>Message GetChargingProfilesResponse</p> <ul style="list-style-type: none">- status must be <i>Accepted</i> <p>* Step 6:</p> <p>Message ReportChargingProfilesRequest</p> <ul style="list-style-type: none">- requestId must be <i>Same Id as in the GetChargingProfilesRequest in step 4</i>- EVSEId must be <i>0</i>- chargingProfile must be <i><Configured chargingProfile1></i>
<p>Post scenario validations:</p> <ul style="list-style-type: none">- The same profile is reported as send in step 1

TC_K_130_CS: Set Charging Profile - PriorityCharging unsupported

Test case name	Set Charging Profile - PriorityCharging unsupported
Test case Id	TC_K_130_CS
Use case Id(s)	K01
Requirement(s)	K01.FR.120
System under test	Charging Station
Description	Test case to verify that a Charging Station that doesn't support PriorityCharging correctly rejects a SetChargingProfileRequest with chargingProfilePurpose set to PriorityCharging.
Purpose	To verify that the Charging Station correctly handles a SetChargingProfileRequest with an unsupported chargingProfilePurpose (PriorityCharging).
Prerequisite(s)	SmartChargingCtrlr.SupportedAdditionalPurposes doesn't contain <i>PriorityCharging</i>

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>Booted</i>

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetChargingProfileResponse	<p>1. The Test System sends a SetChargingProfileRequest with</p> <p>evseld 0</p> <p>chargingProfile.id 1</p> <p>chargingProfile.stackLevel 0</p> <p>chargingProfile.chargingProfilePurpose <i>PriorityCharging</i></p> <p>chargingProfile.chargingProfileKind <i>Absolute</i></p> <p>chargingProfile.chargingSchedule.chargingRateUnit A</p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod[0].startPeriod 0</p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod[0].limit 16 * <limit multiplier></p> <p>Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></p>

Tool validations
<p>* Step 2:</p> <p>Message SetChargingProfileResponse</p> <ul style="list-style-type: none"> - status must be <i>Rejected</i> - statusInfo.reasonCode must be <i>UnsupportedPurpose</i> or <omitted>
Post scenario validations:
N/a

TC_K_131_CS: Set Charging Profile - LocalGeneration unsupported

Test case name	Set Charging Profile - LocalGeneration unsupported
Test case Id	TC_K_131_CS
Use case Id(s)	K01
Requirement(s)	K01.FR.120
System under test	Charging Station
Description	Test case to verify that a Charging Station that doesn't support LocalGeneration correctly rejects a SetChargingProfileRequest with chargingProfilePurpose set to LocalGeneration.
Purpose	To verify that the Charging Station correctly handles a SetChargingProfileRequest with an unsupported chargingProfilePurpose (LocalGeneration).
Prerequisite(s)	SmartChargingCtrlr.SupportedAdditionalPurposes doesn't contain <i>LocalGeneration</i>

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>Booted</i>

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetChargingProfileResponse	<p>1. The Test System sends a SetChargingProfileRequest with</p> <p>evseld 0</p> <p>chargingProfile.id 1</p> <p>chargingProfile.stackLevel 0</p> <p>chargingProfile.chargingProfilePurpose <i>LocalGeneration</i></p> <p>chargingProfile.chargingProfileKind <i>Absolute</i></p> <p>chargingProfile.chargingSchedule.chargingRateUnit A</p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod[0].startPeriod 0</p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod[0].limit 6 * <limit multiplier></p> <p>Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></p>

Tool validations
<p>* Step 2:</p> <p>Message SetChargingProfileResponse</p> <ul style="list-style-type: none"> - status must be <i>Rejected</i> - statusInfo.reasonCode must be <i>UnsupportedPurpose</i> or <omitted>
Post scenario validations:
N/a

TC_K_132_CS: Set Charging Profile - useLocalTime unsupported

Test case name	Set Charging Profile - useLocalTime unsupported
Test case Id	TC_K_132_CS
Use case Id(s)	K01
Requirement(s)	K01.FR.123
System under test	Charging Station
Description	Test case to verify that a Charging Station that doesn't support useLocalTime correctly rejects a SetChargingProfileRequest with useLocalTime set to true.
Purpose	To verify that the Charging Station correctly handles a SetChargingProfileRequest with an unsupported feature (useLocalTime).
Prerequisite(s)	SmartChargingCtrlr.SupportsFeature[UseLocalTime] is <i>false</i> or absent

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>Booted</i>

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetChargingProfileResponse	1. The Test System sends a SetChargingProfileRequest with evseld 0 chargingProfile.id 1 chargingProfile.stackLevel 0 chargingProfile.chargingProfilePurpose <i>TxProfile</i> chargingProfile.chargingProfileKind <i>Absolute</i> chargingProfile.chargingSchedule.chargingRateUnit <i>A</i> chargingProfile.chargingSchedule.chargingSchedulePeriod[0].startPeriod 0 chargingProfile.chargingSchedule.chargingSchedulePeriod[0].limit 6 * <i><limit multiplier></i> <i>Note: Check Determine Charging Profile Limit Multiplier for <i><limit multiplier></i></i> chargingProfile.chargingSchedule.useLocalTime <i>true</i>

Tool validations
* Step 2: Message SetChargingProfileResponse - status must be <i>Rejected</i> - statusInfo.reasonCode must be <i>InvalidSchedule</i> or <i><omitted></i>
Post scenario validations: N/a

TC_K_133_CS: Set Charging Profile - RandomizedDelay unsupported

Test case name	Set Charging Profile - RandomizedDelay unsupported
Test case Id	TC_K_133_CS
Use case Id(s)	K01
Requirement(s)	K01.FR.124
System under test	Charging Station
Description	Test case to verify that a Charging Station that doesn't support RandomizedDelay correctly rejects a SetChargingProfileRequest with randomizedDelay > 0.
Purpose	To verify that the Charging Station correctly handles a SetChargingProfileRequest with an unsupported feature (RandomizedDelay).
Prerequisite(s)	SmartChargingCtrlr.SupportsFeature[RandomizedDelay] is <i>false</i> or absent

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>Booted</i>

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetChargingProfileResponse	<p>1. The Test System sends a SetChargingProfileRequest with</p> <p>evseld 0</p> <p>chargingProfile.id 1</p> <p>chargingProfile.stackLevel 0</p> <p>chargingProfile.chargingProfilePurpose TxProfile</p> <p>chargingProfile.chargingProfileKind Absolute</p> <p>chargingProfile.chargingSchedule.chargingRateUnit A</p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod[0].startPeriod 0</p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod[0].limit 6 * <limit multiplier></p> <p>Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></p> <p>chargingProfile.chargingSchedule.randomizedDelay 10</p>

Tool validations
<p>* Step 2:</p> <p>Message SetChargingProfileResponse</p> <ul style="list-style-type: none"> - status must be <i>Rejected</i> - statusInfo.reasonCode must be <i>InvalidSchedule</i> or <omitted>
Post scenario validations: N/a

TC_K_134_CS: Set Charging Profile - LimitAtSoC unsupported

Test case name	Set Charging Profile - LimitAtSoC unsupported
Test case Id	TC_K_134_CS
Use case Id(s)	K01
Requirement(s)	K01.FR.125
System under test	Charging Station
Description	Test case to verify that a Charging Station that doesn't support LimitAtSoC correctly rejects a SetChargingProfileRequest with limitAtSoC field.
Purpose	To verify that the Charging Station correctly handles a SetChargingProfileRequest with an unsupported feature (LimitAtSoC).
Prerequisite(s)	SmartChargingCtrlr.SupportsFeature[LimitAtSoC] is <i>false</i> or absent

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>Booted</i>

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetChargingProfileResponse	<p>1. The Test System sends a SetChargingProfileRequest with</p> <p>evseld 0</p> <p>chargingProfile.id 1</p> <p>chargingProfile.stackLevel 0</p> <p>chargingProfile.chargingProfilePurpose TxProfile</p> <p>chargingProfile.chargingProfileKind Absolute</p> <p>chargingProfile.chargingSchedule.chargingRateUnit A</p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod[0].startPeriod 0</p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod[0].limit 6 * <limit multiplier></p> <p>Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></p> <p>chargingProfile.chargingSchedule.limitAtSoC.soc 80</p> <p>chargingProfile.chargingSchedule.limitAtSoC.limit 10 * <limit multiplier></p> <p>Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></p>

Tool validations
<p>* Step 2:</p> <p>Message SetChargingProfileResponse</p> <ul style="list-style-type: none"> - status must be <i>Rejected</i> - statusInfo.reasonCode must be <i>InvalidSchedule</i> or <omitted>
Post scenario validations:
N/a

TC_K_135_CS: Idle operationMode - Set Charging Profile - EvseSleep unsupported

Test case name	Idle operationMode - Set Charging Profile - EvseSleep unsupported
Test case Id	TC_K_135_CS
Use case Id(s)	K01
Requirement(s)	K01.FR.126
System under test	Charging Station
Description	Test case to verify that a Charging Station that doesn't support EvseSleep correctly rejects a SetChargingProfileRequest with evseSleep set to true.
Purpose	To verify that the Charging Station correctly handles a SetChargingProfileRequest with an unsupported feature (EvseSleep).
Prerequisite(s)	SmartChargingCtrlr.SupportsFeature[EvseSleep] is <i>false</i> or absent

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>Booted</i>

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetChargingProfileResponse	<p>1. The Test System sends a SetChargingProfileRequest with</p> <p>evseId 0</p> <p>chargingProfile.id 1</p> <p>chargingProfile.stackLevel 0</p> <p>chargingProfile.chargingProfilePurpose <i>TxProfile</i></p> <p>chargingProfile.chargingProfileKind <i>Absolute</i></p> <p>chargingProfile.chargingSchedule.chargingRateUnit A</p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod[0].startPeriod 0</p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod[0].limit 6 * <limit multiplier></p> <p>Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod[0].evseSleep true</p>

Tool validations
<p>* Step 2:</p> <p>Message SetChargingProfileResponse</p> <ul style="list-style-type: none"> - status must be <i>Rejected</i> - statusInfo.reasonCode must be <i>InvalidSchedule</i> or <omitted>
Post scenario validations: N/a

TC_K_136_CS: Set Charging Profile - Local time - TimeZone

Test case name	Set Charging Profile - Local time - TimeZone
Test case Id	TC_K_103_CS
Use case Id(s)	K01
Requirement(s)	K01.FR.90
System under test	Charging Station
Description	To enable the CSMS to set charging profiles with local times so the charging power can be influenced on local time base independent on DST.
Purpose	To verify if the Charging station is able to calculate the Zulu time from a chargingProfileSchedule using local times.
Prerequisite(s)	<ul style="list-style-type: none"> - SmartChargingCtrlr.SupportsFeature[UseLocalTime] is true - ClockCtrlr.TimeZone is present

Before (Preparations)
Configuration State: - ClockCtrlr.TimeZone is "" (if present)
Memory State: N/a
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with SetVariablesResponse	1. Test System sends SetVariablesRequest with: <ul style="list-style-type: none"> - setVariableData[0].variable.name = <i>TimeOffset</i> - setVariableData[0].component.name = <i>ClockCtrlr</i> - setVariableData[0].attributeValue = "" - setVariableData[0].attributeType = <i>Actual</i> - setVariableData[1].variable.name = <i>TimeZone</i> - setVariableData[1].component.name = <i>ClockCtrlr</i> - setVariableData[1].attributeValue = <i>UTC+2:00</i> - setVariableData[1].attributeType = <i>Actual</i>
4. The Charging Station responds with a SetChargingProfileResponse	3. The Test System sends a SetChargingProfileRequest with <ul style="list-style-type: none"> evseld 0 chargingProfile.id <Configured chargingProfileId> chargingProfile.chargingProfilePurpose <i>TxDefaultProfile</i> chargingProfile.chargingSchedule.useLocalTime <i>true</i> chargingProfile.chargingSchedule.duration <Configured duration> chargingProfile.chargingSchedule.startSchedule <current dateTime + 02:10> chargingProfile.chargingSchedule.chargingRateUnit <Configured chargingRateUnit> chargingProfile.chargingSchedule.chargingSchedulePeriod.startPeriod 0 chargingProfile.chargingSchedule.chargingSchedulePeriod.limit 6 * <limit multiplier> Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier> chargingProfile.chargingSchedule.chargingSchedulePeriod.numberPhases <Configured numberPhases>

Main (Test scenario)	
6. The Charging Station responds with a GetCompositeScheduleResponse	5. The Test System sends a GetCompositeScheduleRequest with evseld 0 duration is 900 chargingRateUnit <Configured chargingRateUnit>

Tool validations
<p>* Step 2:</p> <p>Message: SetVariablesResponse</p> <ul style="list-style-type: none">- setVariableResult[1].attributeStatus must be <i>Accepted</i>- setVariableResult[1].component.name = <i>ClockCtrlr</i>- setVariableResult[1].variable.name = <i>TimeZone</i> <p>* Step 4:</p> <p>Message SetChargingProfileResponse</p> <ul style="list-style-type: none">- status must be <i>Accepted</i> <p>* Step 6:</p> <p>Message: GetCompositeScheduleResponse</p> <p>Note: The period of time between the <i>scheduleStart</i> from the <i>SetChargingProfileRequest</i> and the <i>scheduleStart</i> from the <i>GetCompositeScheduleResponse</i> is called <i>x</i>.</p> <ul style="list-style-type: none">- status must be <i>Accepted</i>- schedule.evseld must be 0- schedule.scheduleStart must be <current dateTime +/- max deviation>- schedule.chargingSchedulePeriod[0].startPeriod 0- schedule.chargingSchedulePeriod[1].startPeriod 600 - <i>x</i>- schedule.chargingSchedulePeriod[1].limit 6 * <limit multiplier> <p>Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></p>
<p>Post scenario validations:</p> <p>N/a</p>

L Firmware Management

TC_L_01_CS: Secure Firmware Update - Installation successful

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 Test System 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 <i>Downloading</i>	4. The Test System responds with a FirmwareStatusNotificationResponse
5. The Charging Station sends a FirmwareStatusNotificationRequest With status <i>Downloaded</i>	6. The Test System responds with a FirmwareStatusNotificationResponse
7. The Charging Station sends a FirmwareStatusNotificationRequest With status <i>SignatureVerified</i>	8. The Test System responds with a FirmwareStatusNotificationResponse
9. The Charging Station notifies the CSMS about the current state of all connectors. <u>Note(s):</u> - This step is optional. The Charging Station may wants to set its connectors to <i>Unavailable</i> , before proceeding installing the new firmware.	10. The Test System responds accordingly.
11. Execute Reusable State <i>RebootBeforeFirmwareInstallation</i>	
<u>Note:</u> This step only needs to be executed if the Charging Station needs to reboot before firmware <u>installation</u> .	

Main (Test scenario)	
<p>12. The Charging Station sends a FirmwareStatusNotificationRequest With status <i>Installing</i></p> <p><u>Note(s):</u> - This step only needs to be executed if the Charging Station did NOT reboot before firmware <u>installation</u>, at step 11.</p>	<p>13. The Test System responds with a FirmwareStatusNotificationResponse</p>
<p>14. Execute Reusable State <i>RebootBeforeFirmwareActivation</i></p> <p><u>Note:</u> This step only needs to be executed if the Charging Station needs to reboot before firmware <u>activation</u>.</p>	
<p>15. The Test System waits for the Charging Station to reconnect.</p> <p><u>Note:</u> 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.</p> <p><u>Note:</u> Step 16 through 21 can be send in a different order.</p>	
<p>16. The Charging Station notifies the CSMS about the current state of all connectors.</p> <p><u>Note(s):</u> - This step only needs to be executed if the connectors were previously set to <i>Unavailable</i> (at step 9) and the Charging Station did not report setting them back to <i>Available</i> (after a reboot sequence at step 11 or 14) yet.</p>	<p>17. The Test System responds accordingly.</p>
<p>18. The Charging Station sends a FirmwareStatusNotificationRequest With status <i>Installed</i></p>	<p>19. The Test System responds with a FirmwareStatusNotificationResponse</p>
<p>20. The Charging Station sends a SecurityEventNotificationRequest With type <i>FirmwareUpdated</i></p>	<p>21. The Test System responds with a SecurityEventNotificationResponse</p>

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

TC_L_02_CS: Secure Firmware Update - InstallScheduled

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 <code>UpdateFirmwareRequest</code> with a <code>signingCertificate</code> .
Purpose	To verify if the Charging Station is able securely download a new firmware and schedule its installation.
Prerequisite(s)	<ul style="list-style-type: none"> - A file server has been setup according to the (by the Charging Station) supported file transfer protocol(s), indicated by the configuration variable FileTransferProtocols. - The Test System configuration firmware <code>installDateTime</code> needs to set to a future <code>dateTime</code>.

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 Test System sends a UpdateFirmwareRequest with firmware.location <Configured <i>firmware_location</i> > firmware.retrieveDateTime <Current <i>DateTime</i> - 2 hours> firmware.signingCertificate <Configured <i>signingCertificate</i> > firmware.signature <Configured <i>signature</i> > firmware.installDateTime <Current <i>DateTime</i> + <Configured <i>Install Offset Period</i> >>
3. The Charging Station sends a FirmwareStatusNotificationRequest With status <i>Downloading</i>	4. The Test System responds with a FirmwareStatusNotificationResponse
5. The Charging Station sends a FirmwareStatusNotificationRequest With status <i>Downloaded</i>	6. The Test System responds with a FirmwareStatusNotificationResponse
7. The Charging Station sends a FirmwareStatusNotificationRequest With status <i>SignatureVerified</i>	8. The Test System responds with a FirmwareStatusNotificationResponse
9. The Charging Station sends a FirmwareStatusNotificationRequest With status <i>InstallScheduled</i>	10. The Test System responds with a FirmwareStatusNotificationResponse
<u>Note(s):</u> - The Charging Station will start installing the firmware after the set <code>installDateTime</code> is reached.	
11. The Charging Station notifies the CSMS about the current state of all connectors.	12. The Test System responds accordingly.
<u>Note(s):</u> - This step is optional. The Charging Station may wants to set its connectors to <i>Unavailable</i> , before proceeding installing the new firmware.	

Main (Test scenario)	
13. Execute Reusable State <i>RebootBeforeFirmwareInstallation</i>	
<u>Note:</u> This step only needs to be executed if the Charging Station needs to reboot before firmware <u>installation</u> .	
14. The Charging Station sends a FirmwareStatusNotificationRequest With status <i>Installing</i>	15. The Test System responds with a FirmwareStatusNotificationResponse
<u>Note(s):</u> - This step only needs to be executed if the Charging Station did NOT reboot before firmware <u>installation</u> , at step 13.	
16. Execute Reusable State <i>RebootBeforeFirmwareActivation</i>	
<u>Note:</u> This step only needs to be executed if the Charging Station needs to reboot before firmware <u>activation</u> .	
17. The Test System waits for the Charging Station to reconnect.	
<u>Note:</u> 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.	
<u>Note:</u> 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.	19. The Test System responds accordingly.
<u>Note(s):</u> - This step only needs to be executed if the connectors were previously set to <i>Unavailable</i> (at step 11) and the Charging Station did not report setting them back to <i>Available</i> (after a reboot sequence at step 13 or 16) yet.	
20. The Charging Station sends a FirmwareStatusNotificationRequest With status <i>Installed</i>	21. The Test System responds with a FirmwareStatusNotificationResponse
22. The Charging Station sends a SecurityEventNotificationRequest With type <i>FirmwareUpdated</i>	23. The Test System responds with a SecurityEventNotificationResponse

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

TC_L_03_CS: Secure Firmware Update - DownloadScheduled

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 Test System 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 Test System 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 <u>Note(s):</u> - The Charging Station will start downloading the firmware after the set retrieveDateTime is reached.	4. The Test System responds with a FirmwareStatusNotificationResponse
5. The Charging Station sends a FirmwareStatusNotificationRequest With status Downloading	6. The Test System responds with a FirmwareStatusNotificationResponse
7. The Charging Station sends a FirmwareStatusNotificationRequest With status Downloaded	8. The Test System responds with a FirmwareStatusNotificationResponse
9. The Charging Station sends a FirmwareStatusNotificationRequest With status SignatureVerified	10. The Test System responds with a FirmwareStatusNotificationResponse
11. The Charging Station notifies the CSMS about the current state of all connectors. <u>Note(s):</u> - This step is optional. The Charging Station may wants to set its connectors to Unavailable, before proceeding installing the new firmware.	12. The Test System responds accordingly.

Main (Test scenario)	
13. Execute Reusable State <i>RebootBeforeFirmwareInstallation</i>	
<u>Note:</u> This step only needs to be executed if the Charging Station needs to reboot before firmware <u>installation</u> .	
14. The Charging Station sends a FirmwareStatusNotificationRequest With status <i>Installing</i>	15. The Test System responds with a FirmwareStatusNotificationResponse
<u>Note(s):</u> - This step only needs to be executed if the Charging Station did NOT reboot before firmware <u>installation</u> , at step 13.	
16. Execute Reusable State <i>RebootBeforeFirmwareActivation</i>	
<u>Note:</u> This step only needs to be executed if the Charging Station needs to reboot before firmware <u>activation</u> .	
17. The Test System waits for the Charging Station to reconnect.	
<u>Note:</u> 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.	
<u>Note:</u> 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.	19. The Test System responds accordingly.
<u>Note(s):</u> - This step only needs to be executed if the connectors were previously set to <i>Unavailable</i> (at step 11) and the Charging Station did not report setting them back to <i>Available</i> (after a reboot sequence at step 13 or 16) yet.	
20. The Charging Station sends a FirmwareStatusNotificationRequest With status <i>Installed</i>	21. The Test System responds with a FirmwareStatusNotificationResponse
22. The Charging Station sends a SecurityEventNotificationRequest With type <i>FirmwareUpdated</i>	23. The Test System responds with a SecurityEventNotificationResponse

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

TC_L_05_CS: Secure Firmware Update - InvalidCertificate

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 Test System 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 Test System 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

TC_L_06_CS: Secure Firmware Update - InvalidSignature

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 Test System 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 <i>Downloading</i>	4. The Test System responds with a FirmwareStatusNotificationResponse .
5. The Charging Station sends a FirmwareStatusNotificationRequest . With status <i>Downloaded</i>	6. The Test System responds with a FirmwareStatusNotificationResponse .
<u>Note:</u> Step 7 through 10 can be sent in a different order.	
7. The Charging Station sends a FirmwareStatusNotificationRequest . With status <i>InvalidSignature</i>	8. The Test System responds with a FirmwareStatusNotificationResponse .
9. The Charging Station sends a SecurityEventNotificationRequest . With type <i>InvalidFirmwareSignature</i>	10. The Test System responds with a SecurityEventNotificationResponse .

Tool validations
<div>* Step 2: Message UpdateFirmwareResponse - status <i>Accepted</i></div> <div>* Step 3: Message FirmwareStatusNotificationRequest - status <i>Downloading</i></div> <div>* Step 5: Message FirmwareStatusNotificationRequest - status <i>Downloaded</i></div> <div>* Step 7: Message FirmwareStatusNotificationRequest - status <i>InvalidSignature</i></div> <div>* Step 9: Message SecurityEventNotificationRequest - type <i>InvalidFirmwareSignature</i></div>
<div>Post scenario validations: N/a</div>

TC_L_07_CS: Secure Firmware Update - DownloadFailed

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 Test System 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 Test System 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 <i>Downloading</i> <u>Note(s):</u> - This step is optional. The Charging Station may immediately identify downloading the firmware is not possible.	4. The Test System responds with a FirmwareStatusNotificationResponse .
5. The Charging Station sends a FirmwareStatusNotificationRequest . With status <i>DownloadFailed</i>	6. The Test System responds with a FirmwareStatusNotificationResponse .

Tool validations

* Step 2:

Message **UpdateFirmwareResponse**- **status** *Accepted*

* Step 3:

Message **FirmwareStatusNotificationRequest**- **status** *Downloading*

* Step 5:

Message **FirmwareStatusNotificationRequest**- **status** *DownloadFailed*

Tool validations
Post scenario validations: N/a

TC_L_08_CS: Secure Firmware Update - InstallVerificationFailed or InstallationFailed

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 Test System 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 Test System 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 <i>Downloading</i>	4. The Test System responds with a FirmwareStatusNotificationResponse
5. The Charging Station sends a FirmwareStatusNotificationRequest With status <i>Downloaded</i>	6. The Test System responds with a FirmwareStatusNotificationResponse
7. The Charging Station sends a FirmwareStatusNotificationRequest With status <i>SignatureVerified</i>	8. The Test System responds with a FirmwareStatusNotificationResponse
9. The Charging Station notifies the CSMS about the current state of all connectors.	10. The Test System responds accordingly.
Note(s): - This step is optional. The Charging Station may wants to set its connectors to Unavailable, before proceeding installing the new firmware.	

Main (Test scenario)	
11. Execute Reusable State <i>RebootBeforeFirmwareInstallation</i>	
<i>Note: This step only needs to be executed if the Charging Station needs to reboot before firmware <u>installation</u>.</i>	
12. The Charging Station sends a FirmwareStatusNotificationRequest With status <i>Installing</i>	13. The Test System responds with a FirmwareStatusNotificationResponse
<i>Note(s):</i> - This step only needs to be executed if the Charging Station did NOT reboot before firmware <u>installation</u> , at step 11.	
<i>Note: Step 14 through 17 can be send in a different order.</i>	
14. The Charging Station notifies the CSMS about the current state of all connectors.	15. The Test System responds accordingly.
<i>Note(s):</i> - This step only needs to be executed if the connectors were previously set to <i>Unavailable</i> (at step 9) and the Charging Station did not report setting them back to <i>Available</i> (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.	
16. The Charging Station sends a FirmwareStatusNotificationRequest With status <i>InstallVerificationFailed</i> or <i>InstallationFailed</i>	17. The Test System responds with a FirmwareStatusNotificationResponse

Tool validations
<p>* Step 2: Message UpdateFirmwareResponse - status <i>Accepted</i></p> <p>* Step 3: Message FirmwareStatusNotificationRequest - status <i>Downloading</i></p> <p>* Step 5: Message FirmwareStatusNotificationRequest - status <i>Downloaded</i></p> <p>* Step 7: Message FirmwareStatusNotificationRequest - status <i>SignatureVerified</i></p> <p>* Step 9: Message: StatusNotificationRequest - connectorStatus <i>Unavailable</i></p> <p>Or</p> <p>Message: NotifyEventRequest - eventData[0].trigger <i>Delta</i> - eventData[0].actualValue <i>"Unavailable"</i> - eventData[0].component.name <i>"Connector"</i> - eventData[0].variable.name <i>"AvailabilityState"</i></p> <p>* Step 12: Message FirmwareStatusNotificationRequest - status <i>Installing</i></p> <p>* Step 14: Message: StatusNotificationRequest - connectorStatus <i>Available</i></p> <p>Or</p> <p>Message: NotifyEventRequest - eventData[0].trigger <i>Delta</i> - eventData[0].actualValue <i>"Available"</i> - eventData[0].component.name <i>"Connector"</i> - eventData[0].variable.name <i>"AvailabilityState"</i></p> <p>* Step 16: Message FirmwareStatusNotificationRequest - status <i>InstallVerificationFailed</i> or <i>InstallationFailed</i></p>
<p>Post scenario validations: N/a</p>

TC_L_10_CS: Secure Firmware Update - AcceptedCanceled

Test case name	Secure Firmware Update - AcceptedCanceled
Test case Id	TC_L_10_CS
Use case Id(s)	L01
Requirement(s)	L01.FR.01,L01.FR.10,L01.FR.20,L01.FR.24
System under test	Charging Station
Description	The CSMS is able to request the Charging Station to securely download and install a new firmware by sending an UpdateFirmwareRequest with a signingCertificate.
Purpose	To verify if the Charging Station is able to cancel an ongoing firmware update and start a new one, when receiving an UpdateFirmwareRequest from the CSMS.
Prerequisite(s)	<ul style="list-style-type: none"> - A file server has been setup according to the (by the Charging Station) supported file transfer protocol(s), indicated by the configuration variable FileTransferProtocols. - The Charging Station is able to cancel an ongoing firmware update while it is busy downloading a new firmware file.

Before (Preparations)

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 Test System sends a UpdateFirmwareRequest with requestId = <#1> 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 requestId <#1> and status Downloading	4. The Test System responds with a FirmwareStatusNotificationResponse
6. The Charging Station responds with a UpdateFirmwareResponse With requestId <#1> and status AcceptedCanceled	5. The Test System sends a UpdateFirmwareRequest with requestId = <#2> 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 requestId <#2> and status Downloading	8. The Test System responds with a FirmwareStatusNotificationResponse
9. The Charging Station sends a FirmwareStatusNotificationRequest With status Downloaded	10. The Test System responds with a FirmwareStatusNotificationResponse
11. The Charging Station sends a FirmwareStatusNotificationRequest With status SignatureVerified	12. The Test System responds with a FirmwareStatusNotificationResponse

Main (Test scenario)	
13. The Charging Station notifies the CSMS about the current state of all connectors. <u>Note(s):</u> - This step is optional. The Charging Station may want to set its connectors to <i>Unavailable</i> , before proceeding installing the new firmware.	14. The Test System responds accordingly.
15. Execute Reusable State <i>RebootBeforeFirmwareInstallation</i> <u>Note:</u> This step only needs to be executed if the Charging Station needs to reboot before firmware <u>installation</u> .	
16. The Charging Station sends a FirmwareStatusNotificationRequest With status <i>Installing</i> <u>Note(s):</u> - This step only needs to be executed if the Charging Station did NOT reboot before firmware <u>installation</u> , at step 15.	17. The Test System responds with a FirmwareStatusNotificationResponse
18. Execute Reusable State <i>RebootBeforeFirmwareActivation</i> <u>Note:</u> This step only needs to be executed if the Charging Station needs to reboot before firmware <u>activation</u> .	
19. The Test System waits for the Charging Station to reconnect. <u>Note:</u> 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. <u>Note:</u> Step 20 through 25 can be sent in a different order.	
20. The Charging Station notifies the CSMS about the current state of all connectors. <u>Note(s):</u> - This step only needs to be executed if the connectors were previously set to <i>Unavailable</i> (at step 13) and the Charging Station did not report setting them back to <i>Available</i> (after a reboot sequence at step 15 or 18) yet.	21. The Test System responds accordingly.
22. The Charging Station sends a FirmwareStatusNotificationRequest With status <i>Installed</i>	23. The Test System responds with a FirmwareStatusNotificationResponse
24. The Charging Station sends a SecurityEventNotificationRequest With type <i>FirmwareUpdated</i>	25. The Test System responds with a SecurityEventNotificationResponse

Tool validations
<p>* Step 2: Message UpdateFirmwareResponse - status <i>Accepted</i></p> <p>* Step 3: Message FirmwareStatusNotificationRequest - status <i>Downloading</i> - requestId = <#1></p> <p>* Step 6: Message UpdateFirmwareResponse - status <i>AcceptedCanceled</i> A FirmwareStatusNotificationRequest <i>DownloadFailed</i> or <i>InstallationFailed</i> may be sent for requestId <#1> before or after step 6.</p> <p>(The requestId at the FirmwareStatusNotificationRequest messages must refer to the id <#2> from the second UpdateFirmwareRequest from this point on)</p> <p>* Step 7: Message FirmwareStatusNotificationRequest - status <i>Downloading</i></p> <p>* Step 9: Message FirmwareStatusNotificationRequest - status <i>Downloaded</i></p> <p>* Step 11: Message FirmwareStatusNotificationRequest - status <i>SignatureVerified</i></p> <p>* Step 13: Message: StatusNotificationRequest - connectorStatus <i>Unavailable</i></p> <p>Or Message: NotifyEventRequest - eventData[0].trigger <i>Delta</i> - eventData[0].actualValue <i>"Unavailable"</i> - eventData[0].component.name <i>"Connector"</i> - eventData[0].variable.name <i>"AvailabilityState"</i></p> <p>* Step 16: Message FirmwareStatusNotificationRequest - status <i>Installing</i></p> <p>* Step 20: Message: StatusNotificationRequest - connectorStatus <i>Available</i></p> <p>Or Message: NotifyEventRequest - eventData[0].trigger <i>Delta</i> - eventData[0].actualValue <i>"Available"</i> - eventData[0].component.name <i>"Connector"</i> - eventData[0].variable.name <i>"AvailabilityState"</i></p> <p>* Step 22: Message FirmwareStatusNotificationRequest - status <i>Installed</i></p> <p>* Step 24: Message SecurityEventNotificationRequest - type <i>FirmwareUpdated</i></p>
<p>Post scenario validations: N/a</p>

TC_L_11_CS: Secure Firmware Update - Unable to cancel

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 <code>UpdateFirmwareRequest</code> with a <code>signingCertificate</code> .
Purpose	To verify if the Charging Station is able to reject a firmware update request when it is unable to cancel an ongoing firmware update.
Prerequisite(s)	<ul style="list-style-type: none"> - A file server has been setup according to the (by the Charging Station) supported file transfer protocol(s), indicated by the configuration variable FileTransferProtocols. - The Charging Station is NOT able to cancel an ongoing firmware update.

Before (Preparations)

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 <i>Accepted</i>	1. The Test System sends a UpdateFirmwareRequest with firmware.installDateTime <i><Current DateTime - 2 hours></i> firmware.location <i><Configured firmware_location></i> firmware.retrieveDateTime <i><Current DateTime - 2 hours></i> firmware.signingCertificate <i><Configured signingCertificate></i> firmware.signature <i><Configured signature></i>
3. The Charging Station sends a FirmwareStatusNotificationRequest With status <i>Downloading</i>	4. The Test System responds with a FirmwareStatusNotificationResponse
6. The Charging Station responds with a UpdateFirmwareResponse With status <i>Rejected</i>	5. The Test System sends a UpdateFirmwareRequest with firmware.installDateTime <i><Current DateTime - 2 hours></i> firmware.location <i><Configured firmware_location></i> firmware.retrieveDateTime <i><Current DateTime - 2 hours></i> firmware.signingCertificate <i><Configured signingCertificate></i> firmware.signature <i><Configured signature></i>
7. The Charging Station sends a FirmwareStatusNotificationRequest With status <i>Downloaded</i>	8. The Test System responds with a FirmwareStatusNotificationResponse
9. The Charging Station sends a FirmwareStatusNotificationRequest With status <i>SignatureVerified</i>	10. The Test System responds with a FirmwareStatusNotificationResponse
11. The Charging Station notifies the CSMS about the current state of all connectors. <u>Note(s):</u> - This step is optional. The Charging Station may wants to set its connectors to <i>Unavailable</i> , before proceeding installing the new firmware.	12. The Test System responds accordingly.

Main (Test scenario)	
13. Execute Reusable State <i>RebootBeforeFirmwareInstallation</i>	
<u>Note:</u> <i>This step only needs to be executed if the Charging Station needs to reboot before firmware <u>installation</u>.</i>	
14. The Charging Station sends a FirmwareStatusNotificationRequest With status <i>Installing</i>	15. The Test System responds with a FirmwareStatusNotificationResponse
<u>Note(s):</u> <i>- This step only needs to be executed if the Charging Station did NOT reboot before firmware <u>installation</u>, at step 13.</i>	
16. Execute Reusable State <i>RebootBeforeFirmwareActivation</i>	
<u>Note:</u> <i>This step only needs to be executed if the Charging Station needs to reboot before firmware <u>activation</u>.</i>	
17. The Test System waits for the Charging Station to reconnect.	
<u>Note:</u> <i>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.</i>	
<u>Note:</u> <i>Step 18 through 23 can be send in a different order.</i>	
18. The Charging Station notifies the CSMS about the current state of all connectors.	19. The Test System responds accordingly.
<u>Note(s):</u> <i>- 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.</i>	
20. The Charging Station sends a FirmwareStatusNotificationRequest With status <i>Installed</i>	21. The Test System responds with a FirmwareStatusNotificationResponse
22. The Charging Station sends a SecurityEventNotificationRequest With type <i>FirmwareUpdated</i>	23. The Test System responds with a SecurityEventNotificationResponse

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

TC_L_12_CS: Secure Firmware Update - Unable to download/install firmware with ongoing transaction - AllowNewSessionsPendingFirmwareUpdate is true

Test case name	Secure Firmware Update - Unable to download/install firmware with ongoing transaction - AllowNewSessionsPendingFirmwareUpdate is true
Test case Id	TC_L_12_CS
Use case Id(s)	L01
Requirement(s)	L01.FR.01,L01.FR.06,L01.FR.07,L01.FR.10,L01.FR.20
System under test	Charging Station
Description	The CSMS is able to request the Charging Station to securely download and install a new firmware by sending an UpdateFirmwareRequest with a signingCertificate.
Purpose	To verify if the Charging Station is able to keep allowing new transactions when requested to update the firmware, while there is an ongoing transaction.
Prerequisite(s)	<ul style="list-style-type: none"> - A file server has been setup according to the (by the Charging Station) supported file transfer protocol(s), indicated by the configuration variable FileTransferProtocols. - The Charging Station is able to start more than one transaction at a time. - The Charging Station is unable to download AND install firmware while there is an ongoing transaction.

Before (Preparations)
Configuration State: AllowNewSessionsPendingFirmwareUpdate is <i>true</i> (If implemented)
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferStarted</i> for <Configured connectorId>

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a UpdateFirmwareResponse With status <i>Accepted</i>	1. The Test System 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 <i>DownloadScheduled</i>	4. The Test System responds with a FirmwareStatusNotificationResponse
5. Execute Reusable State <i>EnergyTransferStarted</i> 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 <i>ParkingBayUnoccupied</i> 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 <i>ParkingBayUnoccupied</i> 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).	

Main (Test scenario)	
8. The Charging Station sends a FirmwareStatusNotificationRequest With status <i>Downloading</i>	9. The Test System responds with a FirmwareStatusNotificationResponse
10. The Charging Station sends a FirmwareStatusNotificationRequest With status <i>Downloaded</i>	11. The Test System responds with a FirmwareStatusNotificationResponse
12. The Charging Station sends a FirmwareStatusNotificationRequest With status <i>SignatureVerified</i>	13. The Test System responds with a FirmwareStatusNotificationResponse
14. The Charging Station notifies the CSMS about the current state of all connectors. <u>Note(s):</u> - This step is optional. The Charging Station may wants to set its connectors to <i>Unavailable</i> , before proceeding installing the new firmware.	15. The Test System responds accordingly.
16. Execute Reusable State <i>RebootBeforeFirmwareInstallation</i> <u>Note:</u> This step only needs to be executed if the Charging Station needs to reboot before firmware <u>installation</u> .	
17. The Charging Station sends a FirmwareStatusNotificationRequest With status <i>Installing</i> <u>Note(s):</u> - This step only needs to be executed if the Charging Station did NOT reboot before firmware <u>installation</u> , at step 16.	18. The Test System responds with a FirmwareStatusNotificationResponse
19. Execute Reusable State <i>RebootBeforeFirmwareActivation</i> <u>Note:</u> This step only needs to be executed if the Charging Station needs to reboot before firmware <u>activation</u> .	
20. The Test System waits for the Charging Station to reconnect. <u>Note:</u> 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. <u>Note:</u> 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. <u>Note(s):</u> - This step only needs to be executed if the connectors were previously set to <i>Unavailable</i> (at step 14) and the Charging Station did not report setting them back to <i>Available</i> (after a reboot sequence at step 16 or 19) yet.	22. The Test System responds accordingly.
23. The Charging Station sends a FirmwareStatusNotificationRequest With status <i>Installed</i>	24. The Test System responds with a FirmwareStatusNotificationResponse
25. The Charging Station sends a SecurityEventNotificationRequest With type <i>FirmwareUpdated</i>	26. The Test System responds with a SecurityEventNotificationResponse

Tool validations
<p>* Step 2:</p> <p>Message UpdateFirmwareResponse</p> <ul style="list-style-type: none"> - status <i>Accepted</i> <p>* Step 3:</p> <p>Message FirmwareStatusNotificationRequest</p> <ul style="list-style-type: none"> - status <i>DownloadScheduled</i> <p>* Step 8:</p> <p>Message FirmwareStatusNotificationRequest</p> <ul style="list-style-type: none"> - status <i>Downloading</i> <p>* Step 10:</p> <p>Message FirmwareStatusNotificationRequest</p> <ul style="list-style-type: none"> - status <i>Downloaded</i> <p>* Step 12:</p> <p>Message FirmwareStatusNotificationRequest</p> <ul style="list-style-type: none"> - status <i>SignatureVerified</i> <p>* Step 14:</p> <p>Message: StatusNotificationRequest</p> <ul style="list-style-type: none"> - connectorStatus <i>Unavailable</i> <p>Or</p> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none"> - eventData[0].trigger <i>Delta</i> - eventData[0].actualValue <i>"Unavailable"</i> - eventData[0].component.name <i>"Connector"</i> - eventData[0].variable.name <i>"AvailabilityState"</i> <p>* Step 17:</p> <p>Message FirmwareStatusNotificationRequest</p> <ul style="list-style-type: none"> - status <i>Installing</i> <p>* Step 21:</p> <p>Message: StatusNotificationRequest</p> <ul style="list-style-type: none"> - connectorStatus <i>Available</i> <p>Or</p> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none"> - eventData[0].trigger <i>Delta</i> - eventData[0].actualValue <i>"Available"</i> - eventData[0].component.name <i>"Connector"</i> - eventData[0].variable.name <i>"AvailabilityState"</i> <p>* Step 23:</p> <p>Message FirmwareStatusNotificationRequest</p> <ul style="list-style-type: none"> - status <i>Installed</i> <p>* Step 25:</p> <p>Message SecurityEventNotificationRequest</p> <ul style="list-style-type: none"> - type <i>FirmwareUpdated</i>
<p>Post scenario validations:</p> <p>N/a</p>

TC_L_13_CS: Secure Firmware Update - Unable to download/install firmware with ongoing transaction - AllowNewSessionsPendingFirmwareUpdate is false

Test case name	Secure Firmware Update - Unable to download/install firmware with ongoing transaction - AllowNewSessionsPendingFirmwareUpdate is false
Test case Id	TC_L_13_CS
Use case Id(s)	L01
Requirement(s)	L01.FR.01,L01.FR.06,L01.FR.07,L01.FR.10,L01.FR.20
System under test	Charging Station
Description	The CSMS is able to request the Charging Station to securely download and install a new firmware by sending an UpdateFirmwareRequest with a signingCertificate.
Purpose	To verify if the Charging Station is able to set its available connectors to Unavailable when requested to update the firmware, while there is an ongoing transaction.
Prerequisite(s)	<ul style="list-style-type: none"> - A file server has been setup according to the (by the Charging Station) supported file transfer protocol(s), indicated by the configuration variable FileTransferProtocols. - The configuration variable AllowNewSessionsPendingFirmwareUpdate is implemented. - The Charging Station is unable to download AND install firmware while there is an ongoing transaction.

Before (Preparations)
Configuration State: AllowNewSessionsPendingFirmwareUpdate is <i>false</i>
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a UpdateFirmwareResponse With status <i>Accepted</i>	1. The Test System 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 <i>DownloadScheduled</i> <u>Note:</u> 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 Test System responds with a FirmwareStatusNotificationResponse
5. The Charging Station notifies the CSMS about the current state of its Available connector(s). <u>Note(s):</u> - This step needs to be executed for all connectors with <i>AvailabilityState</i> <i>Available</i> .	6. The Test System responds accordingly.
7. Execute Reusable State <i>ParkingBayUnoccupied</i> for <Configured connectorId> <u>Note(s):</u> - 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).	

Main (Test scenario)	
8. The Charging Station sends a FirmwareStatusNotificationRequest With status <i>Downloading</i>	9. The Test System responds with a FirmwareStatusNotificationResponse
10. The Charging Station sends a FirmwareStatusNotificationRequest With status <i>Downloaded</i>	11. The Test System responds with a FirmwareStatusNotificationResponse
12. The Charging Station sends a FirmwareStatusNotificationRequest With status <i>SignatureVerified</i>	13. The Test System responds with a FirmwareStatusNotificationResponse
14. The Charging Station notifies the CSMS about the current state of all connectors. <u>Note(s):</u> - This step is optional. The Charging Station may wants to set its last connector also to <i>Unavailable</i> , before proceeding installing the new firmware.	15. The Test System responds accordingly.
16. Execute Reusable State <i>RebootBeforeFirmwareInstallation</i> <u>Note:</u> This step only needs to be executed if the Charging Station needs to reboot before firmware <u>installation</u> .	
17. The Charging Station sends a FirmwareStatusNotificationRequest With status <i>Installing</i> <u>Note(s):</u> - This step only needs to be executed if the Charging Station did NOT reboot before firmware <u>installation</u> , at step 16.	18. The Test System responds with a FirmwareStatusNotificationResponse
19. Execute Reusable State <i>RebootBeforeFirmwareActivation</i> <u>Note:</u> This step only needs to be executed if the Charging Station needs to reboot before firmware <u>activation</u> .	
20. The Test System waits for the Charging Station to reconnect. <u>Note:</u> 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. <u>Note:</u> 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. <u>Note(s):</u> - This step only needs to be executed if the connectors were previously set to <i>Unavailable</i> (at step 14) and the Charging Station did not report setting them back to <i>Available</i> (after a reboot sequence at step 16 or 19) yet.	22. The Test System responds accordingly.
23. The Charging Station sends a FirmwareStatusNotificationRequest With status <i>Installed</i>	24. The Test System responds with a FirmwareStatusNotificationResponse
25. The Charging Station sends a SecurityEventNotificationRequest With type <i>FirmwareUpdated</i>	26. The Test System responds with a SecurityEventNotificationResponse

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

TC_L_14_CS: Secure Firmware Update - Unable to install and activate firmware with ongoing transaction - AllowNewSessionsPendingFirmwareUpdate is true

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 <i>Installing</i> phase is not possible while a transaction is ongoing, Charging Station will report <i>InstallScheduled</i> and wait for transaction(s) to finish first, else it will immediately report <i>Installing</i> . 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)	<ul style="list-style-type: none"> - A file server has been setup according to the (by the Charging Station) supported file transfer protocol(s), indicated by the configuration variable FileTransferProtocols. - The Charging Station is able to start more than one transaction at a time. - The Charging Station is unable to install and/or activate firmware while there is an ongoing transaction.

Before (Preparations)
Configuration State: AllowNewSessionsPendingFirmwareUpdate is <i>true</i> (If implemented)
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferStarted</i> for EVSEId 1 and ConnectorId 1

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a UpdateFirmwareResponse With status <i>Accepted</i>	1. The Test System 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 <i>Downloading</i>	4. The Test System responds with a FirmwareStatusNotificationResponse
5. The Charging Station sends a FirmwareStatusNotificationRequest With status <i>Downloaded</i>	6. The Test System responds with a FirmwareStatusNotificationResponse
7. The Charging Station sends a FirmwareStatusNotificationRequest With status <i>SignatureVerified</i>	8. The Test System responds with a FirmwareStatusNotificationResponse
9. The Charging Station sends a FirmwareStatusNotificationRequest With status <i>InstallScheduled</i> or status <i>Installing</i>	10. The Test System responds with a FirmwareStatusNotificationResponse
Note(s): - <i>InstallScheduled</i> only applies when Charging Station is not able to install while a transaction is active.	

Main (Test scenario)	
11. Execute Reusable State <i>EnergyTransferStarted</i> for <Configured second Connector>	
<u>Note(s):</u> - It is allowed to start a second transaction while there is a (scheduled) firmware update.	
11a. If Charging Station reported <i>Installing</i> 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 <i>ParkingBayUnoccupied</i> for <Configured connectorId>	
<u>Note(s):</u> - The Charging Station will proceed to this end state. This will cause the first transaction to stop.	
13. Execute Reusable State <i>ParkingBayUnoccupied</i> for <Configured second Connector>	
<u>Note(s):</u> - 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.	15. The Test System responds accordingly.
<u>Note(s):</u> - This step is optional. The Charging Station may want to set its connectors to <i>Unavailable</i> , before proceeding installing the new firmware.	
16. Execute Reusable State <i>RebootBeforeFirmwareInstallation</i>	
<u>Note:</u> This step only needs to be executed if the Charging Station needs to reboot before firmware <u>installation</u> .	
17. The Charging Station sends a FirmwareStatusNotificationRequest With status <i>Installing</i>	18. The Test System responds with a FirmwareStatusNotificationResponse
<u>Note(s):</u> - This step only needs to be executed if the Charging Station did not report <i>Installing</i> at step 9 and did not reboot before firmware <u>installation</u> , at step 16 (because that step already reports <i>Installing</i>).	
19. Execute Reusable State <i>RebootBeforeFirmwareActivation</i>	
<u>Note:</u> This step only needs to be executed if the Charging Station needs to reboot before firmware <u>activation</u> .	
20. The Test System waits for the Charging Station to reconnect.	
<u>Note:</u> 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.	
<u>Note:</u> Step 21 through 26 can be sent in a different order.	

Main (Test scenario)	
<p>21. The Charging Station notifies the CSMS about the current state of all connectors.</p> <p><u>Note(s):</u> - This step only needs to be executed if the connectors were previously set to <i>Unavailable</i> (at step 14) and the Charging Station did not report setting them back to <i>Available</i> (after a reboot sequence at step 16 or 19) yet.</p>	<p>22. The Test System responds accordingly.</p>
<p>23. The Charging Station sends a FirmwareStatusNotificationRequest With status <i>Installed</i></p>	<p>24. The Test System responds with a FirmwareStatusNotificationResponse</p>
<p>25. The Charging Station sends a SecurityEventNotificationRequest With type <i>FirmwareUpdated</i></p>	<p>26. The Test System responds with a SecurityEventNotificationResponse</p>

Tool validations
<p>* Step 2: Message UpdateFirmwareResponse - status <i>Accepted</i></p> <p>* Step 3: Message FirmwareStatusNotificationRequest - status <i>Downloading</i></p> <p>* Step 5: Message FirmwareStatusNotificationRequest - status <i>Downloaded</i></p> <p>* Step 7: Message FirmwareStatusNotificationRequest - status <i>SignatureVerified</i></p> <p>* Step 9: Message FirmwareStatusNotificationRequest - status <i>InstallScheduled</i> or <i>Installing</i></p> <p>* Step 14: (optional) Message: StatusNotificationRequest - connectorStatus <i>Unavailable</i></p> <p>Or Message: NotifyEventRequest - eventData[0].trigger <i>Delta</i> - eventData[0].actualValue <i>"Unavailable"</i> - eventData[0].component.name <i>"Connector"</i> - eventData[0].variable.name <i>"AvailabilityState"</i></p> <p>* Step 17: (optional depending on step 9) Message FirmwareStatusNotificationRequest - status <i>Installing</i></p> <p>* Step 21: Message: StatusNotificationRequest - connectorStatus <i>Available</i></p> <p>Or Message: NotifyEventRequest - eventData[0].trigger <i>Delta</i> - eventData[0].actualValue <i>"Available"</i> - eventData[0].component.name <i>"Connector"</i> - eventData[0].variable.name <i>"AvailabilityState"</i></p> <p>* Step 23: Message FirmwareStatusNotificationRequest - status <i>Installed</i></p> <p>* Step 25: Message SecurityEventNotificationRequest - type <i>FirmwareUpdated</i></p>
<p>Post scenario validations: N/a</p>

TC_L_15_CS: Secure Firmware Update - Unable to install and activate firmware with ongoing transaction - AllowNewSessionsPendingFirmwareUpdate is false

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 <i>Installing</i> phase is not possible while a transaction is ongoing, Charging Station will report <i>InstallScheduled</i> and wait for transaction(s) to finish first, else it will immediately report <i>Installing</i> . 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)	<ul style="list-style-type: none"> - A file server has been setup according to the (by the Charging Station) supported file transfer protocol(s), indicated by the configuration variable FileTransferProtocols. - The configuration variable AllowNewSessionsPendingFirmwareUpdate is implemented. - The Charging Station is unable to install and/or activate firmware while there is an ongoing transaction.

Before (Preparations)
Configuration State: AllowNewSessionsPendingFirmwareUpdate is <i>false</i>
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a UpdateFirmwareResponse With status <i>Accepted</i>	1. The Test System 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 <i>Downloading</i>	4. The Test System responds with a FirmwareStatusNotificationResponse
5. The Charging Station sends a FirmwareStatusNotificationRequest With status <i>Downloaded</i>	6. The Test System responds with a FirmwareStatusNotificationResponse
7. The Charging Station sends a FirmwareStatusNotificationRequest With status <i>SignatureVerified</i>	8. The Test System responds with a FirmwareStatusNotificationResponse
9. The Charging Station sends a FirmwareStatusNotificationRequest With status <i>InstallScheduled</i> or status <i>Installing</i>	10. The Test System responds with a FirmwareStatusNotificationResponse
Note: <i>InstallScheduled</i> 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 <i>installDateTime</i> 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.	

Main (Test scenario)	
<p>11. The Charging Station notifies the CSMS that its Available connector(s) have been set to Unavailable.</p> <p><u>Note(s):</u> - This step needs to be executed for all connectors with AvailabilityState Available.</p>	<p>12. The Test System responds accordingly.</p>
<p>12a. If Charging Station reported <i>Installing</i> in step 9 then wait a while (30-60 s) before continuing with next steps to stop transaction to allow time to install firmware.</p>	
<p>13. Execute Reusable State <i>ParkingBayUnoccupied</i> for <Configured connectorId></p> <p><u>Note(s):</u> - 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).</p>	
<p>14. The Charging Station notifies the CSMS about the current state of all connectors.</p> <p><u>Note(s):</u> - This step is optional. The Charging Station may want to set its last connector to Unavailable, before proceeding installing the new firmware.</p>	<p>15. The Test System responds accordingly.</p>
<p>16. Execute Reusable State <i>RebootBeforeFirmwareInstallation</i></p> <p><u>Note:</u> This step only needs to be executed if the Charging Station needs to reboot before firmware <u>installation</u>.</p>	
<p>17. The Charging Station sends a FirmwareStatusNotificationRequest With status <i>Installing</i></p> <p><u>Note(s):</u> - This step only needs to be executed if the Charging Station did not report <i>Installing</i> at step 9 and did not reboot before firmware <u>installation</u>, at step 16 (because that step already reports <i>Installing</i>).</p>	<p>18. The Test System responds with a FirmwareStatusNotificationResponse</p>
<p>19. Execute Reusable State <i>RebootBeforeFirmwareActivation</i></p> <p><u>Note:</u> This step only needs to be executed if the Charging Station needs to reboot before firmware <u>activation</u>.</p>	
<p>20. The Test System waits for the Charging Station to reconnect.</p> <p><u>Note:</u> 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.</p> <p><u>Note:</u> Step 21 through 26 can be sent in a different order.</p>	
<p>21. The Charging Station notifies the CSMS about the current state of all connectors.</p> <p><u>Note(s):</u> - 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.</p>	<p>22. The Test System responds accordingly.</p>

Main (Test scenario)	
23. The Charging Station sends a FirmwareStatusNotificationRequest With status <i>Installed</i>	24. The Test System responds with a FirmwareStatusNotificationResponse
25. The Charging Station sends a SecurityEventNotificationRequest With type <i>FirmwareUpdated</i>	26. The Test System responds with a SecurityEventNotificationResponse

Tool validations
<p>* Step 2:</p> <p>Message UpdateFirmwareResponse</p> <ul style="list-style-type: none"> - status <i>Accepted</i> <p>* Step 3:</p> <p>Message FirmwareStatusNotificationRequest</p> <ul style="list-style-type: none"> - status <i>Downloading</i> <p>* Step 5:</p> <p>Message FirmwareStatusNotificationRequest</p> <ul style="list-style-type: none"> - status <i>Downloaded</i> <p>* Step 7:</p> <p>Message FirmwareStatusNotificationRequest</p> <ul style="list-style-type: none"> - status <i>SignatureVerified</i> <p>* Step 9:</p> <p>Message FirmwareStatusNotificationRequest</p> <ul style="list-style-type: none"> - status <i>InstallScheduled</i> or <i>Installing</i> <p>* Step 11:</p> <p>Message: StatusNotificationRequest</p> <ul style="list-style-type: none"> - connectorStatus <i>Unavailable</i> <p>Or</p> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none"> - eventData[0].trigger <i>Delta</i> - eventData[0].actualValue <i>"Unavailable"</i> - eventData[0].component.name <i>"Connector"</i> - eventData[0].variable.name <i>"AvailabilityState"</i> <p>* Step 14: (optional)</p> <p>Message: StatusNotificationRequest</p> <ul style="list-style-type: none"> - connectorStatus <i>Unavailable</i> <p>Or</p> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none"> - eventData[0].trigger <i>Delta</i> - eventData[0].actualValue <i>"Unavailable"</i> - eventData[0].component.name <i>"Connector"</i> - eventData[0].variable.name <i>"AvailabilityState"</i> <p>* Step 17: (optional depending on step 9)</p> <p>Message FirmwareStatusNotificationRequest</p> <ul style="list-style-type: none"> - status <i>Installing</i> <p>* Step 21:</p> <p>Message: StatusNotificationRequest</p> <ul style="list-style-type: none"> - connectorStatus <i>Available</i> <p>Or</p> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none"> - eventData[0].trigger <i>Delta</i> - eventData[0].actualValue <i>"Available"</i> - eventData[0].component.name <i>"Connector"</i> - eventData[0].variable.name <i>"AvailabilityState"</i> <p>* Step 23:</p> <p>Message FirmwareStatusNotificationRequest</p> <ul style="list-style-type: none"> - status <i>Installed</i> <p>* Step 25:</p> <p>Message SecurityEventNotificationRequest</p> <ul style="list-style-type: none"> - type <i>FirmwareUpdated</i>
<p>Post scenario validations:</p> <p>N/a</p>

TC_L_16_CS: Secure Firmware Update - Able to update firmware with ongoing transaction

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 <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a UpdateFirmwareResponse	1. The Test System 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 <i>Downloading</i>	4. The Test System responds with a FirmwareStatusNotificationResponse .
5. The Charging Station sends a FirmwareStatusNotificationRequest . With status <i>Downloaded</i>	6. The Test System responds with a FirmwareStatusNotificationResponse .
7. The Charging Station sends a FirmwareStatusNotificationRequest . With status <i>SignatureVerified</i>	8. The Test System responds with a FirmwareStatusNotificationResponse .
9. The Charging Station sends a FirmwareStatusNotificationRequest . With status <i>Installing</i>	10. The Test System responds with a FirmwareStatusNotificationResponse .
11. The Test System waits for the Charging Station to reconnect.	
<u>Note:</u> The Charging Station reconnects to reestablish the protocol version handshake.	
12. The Charging Station sends a FirmwareStatusNotificationRequest . With status <i>Installed</i>	13. The Test System responds with a FirmwareStatusNotificationResponse .
14. The Charging Station sends a SecurityEventNotificationRequest With type <i>FirmwareUpdated</i>	15. The Test System responds with a SecurityEventNotificationResponse

Tool validations
<p>* Step 2: Message UpdateFirmwareResponse - status <i>Accepted</i></p> <p>* Step 3: Message FirmwareStatusNotificationRequest - status <i>Downloading</i></p> <p>* Step 5: Message FirmwareStatusNotificationRequest - status <i>Downloaded</i></p> <p>* Step 7: Message FirmwareStatusNotificationRequest - status <i>SignatureVerified</i></p> <p>* Step 9: Message FirmwareStatusNotificationRequest - status <i>Installing</i></p> <p>* Step 12: Message FirmwareStatusNotificationRequest - status <i>Installed</i></p> <p>* Step 14: Message SecurityEventNotificationRequest - type <i>FirmwareUpdated</i></p>
<p>Post scenario validations: N/a</p>

TC_L_18_CS: Secure Firmware Update - Missing firmware signing certificate and signature

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 Test System 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 <i>Rejected</i> OR <i>InvalidCertificate</i>
Post scenario validations: N/a

M CertificateManagement

TC_M_01_CS: Install CA certificate - CSMSRootCertificate

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 <i>CertificateInstalled</i> for certificateType CSMSRootCertificate (Root 2)	
<u>Note(s):</u> - When the Charging Station has the following configuration; AdditionalRootCertificateCheck implemented with value true , then a custom CSMSRootCertificate should be used.	
2. Execute Reusable State <i>GetInstalledCertificates</i> for certificateType CSMSRootCertificate	

Tool validations
N/a
Post scenario validations: N/a

TC_M_02_CS: Install CA certificate - ManufacturerRootCertificate

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 <i>CertificateInstalled</i> for certificateType <i>ManufacturerRootCertificate</i>	
2. Execute Reusable State <i>GetInstalledCertificates</i> for certificateType <i>ManufacturerRootCertificate</i>	

Tool validations
N/a
Post scenario validations: N/a

TC_M_03_CS: Install CA certificate - V2GRootCertificate

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)	<ul style="list-style-type: none"> - The Charging Station supports ISO 15118. - The Charging Station does NOT have the following configuration; AdditionalRootCertificateCheck is implemented with value <i>true</i>

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

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>CertificateInstalled</i> for certificateType V2GRootCertificate	
2. Execute Reusable State <i>GetInstalledCertificates</i> for certificateType V2GRootCertificate	

Tool validations
N/a
Post scenario validations: N/a

TC_M_04_CS: Install CA certificate - MORootCertificate

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)	<ul style="list-style-type: none"> - The Charging Station supports ISO 15118. - The Charging Station does NOT have the following configuration; AdditionalRootCertificateCheck is implemented with value <i>true</i>

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

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>CertificateInstalled</i> for certificateType MORootCertificate	
2. Execute Reusable State <i>GetInstalledCertificates</i> for certificateType MORootCertificate	

Tool validations
N/a
Post scenario validations: N/a

TC_M_07_CS: Install CA certificate - Rejected - Certificate invalid

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)	<ul style="list-style-type: none"> - The Charging Station supports Security Profile 2 or 3. - The Charging Station does NOT have the following configuration; AdditionalRootCertificateCheck is implemented with value <i>true</i>

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

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a InstallCertificateResponse	1. The Test System sends a InstallCertificateRequest with certificateType is <i>CSMSRootCertificate</i> certificate is <Generated Expired Certificate>
4. The Charging Station responds with a GetInstalledCertificateIdsResponse	3. The Test System sends a GetInstalledCertificateIdsRequest with certificateType is <i>CSMSRootCertificate</i>

Tool validations
<p>* Step 2: Message: InstallCertificateResponse - status must be <i>Rejected</i></p> <p>* Step 4: Message: GetInstalledCertificateIdsResponse - status must be <i>NotFound</i></p> <p>OR</p> <p>- status must be <i>Accepted</i> - certificateHashDataChain must NOT contain an entry with following values: – certificateType is <i>CSMSRootCertificate</i> – certificateHashData contains <HashData from configured new CSMS Root certificate></p>
Post scenario validations: N/a

TC_M_09_CS: Install CA certificate - AdditionalRootCertificateCheck - Rejected

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)	<ul style="list-style-type: none"> - The Charging Station supports Security Profile 2 or 3. - The Charging Station has the configuration variable AdditionalRootCertificateCheck implemented with value <i>true</i>

Before (Preparations)

Configuration State:

N/a

Memory State:

N/a

Reusable State(s):

N/a

Main (Test scenario)

Charging Station	CSMS
2. The Charging Station responds with a InstallCertificateResponse	1. The Test System sends a InstallCertificateRequest with certificateType is <i>CSMSRootCertificate</i> certificate is <i><Configured CSMSRootCertificate></i> <u>Note(s):</u> - <i>CSMSRootCertificate must have not been signed by old certificate.</i>
4. The Charging Station responds with a GetInstalledCertificateIdsResponse	3. The Test System sends a GetInstalledCertificateIdsRequest with certificateType is <i>CSMSRootCertificate</i>

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

TC_M_30_CS: Install CA certificate - AdditionalRootCertificateCheck - Reconnect using new CSMS Root - Success

Test case name	Install CA certificate - AdditionalRootCertificateCheck - Reconnect using new CSMS Root - Success
Test case Id	TC_M_30_CS
Use case Id(s)	M05
Requirement(s)	M05.FR.13
System under test	Charging Station
Description	The CSMS is able to request the Charging Station to install new Root CA certificates using the InstallCertificateRequest message.
Purpose	To verify if the Charging Station is able to reconnect to the CSMS, while using a new CSMS Root certificate.
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station supports Security Profile 2 or 3. - The Charging Station has the configuration variable AdditionalRootCertificateCheck implemented with value <i>true</i> - The at the Test System configured new CSMSRootCertificate must be signed by the old CSMS Root certificate.

Before (Preparations)
Configuration State: N/a
Memory State: <i>CertificateInstalled</i> for certificateType <i>CSMSRootCertificate</i> and certificate <Configured new CSMS Root certificate 2> If security profile 3 is enabled, then: <i>RenewChargingStationCertificate</i> for certificateType <i>ChargingStationCertificate</i>
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a ResetResponse	1. The Test System sends a ResetRequest with type OnIdle
4. During the TLS handshake the Charging Station validates the CSMS certificate.	3. During the TLS handshake the Test System provides a CSMS certificate which is signed by the <Configured new CSMS Root certificate>
Note(s): - This connection attempt must succeed.	
5. Execute Reusable State <i>Booted</i>	
7. The Charging Station responds with a GetInstalledCertificateIdsResponse	6. The Test System sends a GetInstalledCertificateIdsRequest with certificateType is <i>CSMSRootCertificate</i>

Tool validations
* Step 2: Message ResetResponse - status <i>Accepted</i> * Step 7: Message: GetInstalledCertificateIdsResponse - status must be <i>Accepted</i> - certificateHashDataChain must NOT contain an entry with following values: - certificateType is <i>CSMSRootCertificate</i> - certificateHashData contains <HashData from configured old CSMS Root certificate>
Post scenario validations: - N/a

TC_M_31_CS: Install CA certificate - AdditionalRootCertificateCheck - Reconnect using new CSMS Root - Fallback mechanism

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 <i>true</i> - The at the Test System configured new CSMSRootCertificate must be signed by the old CSMS Root certificate.

Before (Preparations)
Configuration State: N/a
Memory State: <i>CertificateInstalled</i> for certificateType <i>CSMSRootCertificate</i> 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 Test System sends a ResetRequest with type OnIdle
4. During the TLS handshake the Charging Station validates the CSMS certificate. <u>Note(s):</u> - 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 Test System provides a CSMS certificate which is signed by the <Configured old CSMS Root certificate>
5. The Charging Station re-validates the CSMS certificate. <u>Note(s):</u> - 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 <i>Booted</i>	
8. The Charging Station responds with a GetInstalledCertificateIdsResponse	7. The Test System sends a GetInstalledCertificateIdsRequest with certificateType is <i>CSMSRootCertificate</i>

Tool validations
<p>* Step 2:</p> <p>Message ResetResponse</p> <ul style="list-style-type: none">- status <i>Accepted</i> <p>* Step 8:</p> <p>Message: GetInstalledCertificateIdsResponse</p> <ul style="list-style-type: none">- status must be <i>Accepted</i>- certificateHashDataChain must contain an entry with following values:- certificateType is <i>CSMSRootCertificate</i>- certificateHashData contains <i><HashData from configured old CSMS Root certificate></i>
<p>Post scenario validations:</p> <ul style="list-style-type: none">- N/a

TC_M_12_CS: Retrieve certificates from Charging Station - CSMSRootCertificate

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: <i>GetInstalledCertificates</i> from certificateType CSMSRootCertificate
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>GetInstalledCertificates</i> for certificateType CSMSRootCertificate	

Tool validations
N/a
Post scenario validations: N/a

TC_M_13_CS: Retrieve certificates from Charging Station - ManufacturerRootCertificate

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 <i>ManufacturerRootCertificate</i>
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>GetInstalledCertificates</i> for certificateType <i>ManufacturerRootCertificate</i>	

Tool validations
N/a
Post scenario validations: N/a

TC_M_14_CS: Retrieve certificates from Charging Station - V2GRootCertificate

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 <i>GetInstalledCertificates</i> for certificateType V2GRootCertificate	

Tool validations
N/a
Post scenario validations: N/a

TC_M_15_CS: Retrieve certificates from Charging Station - V2GCertificateChain

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)	<ul style="list-style-type: none"> - The Charging Station supports ISO 15118. - The Charging Station has atleast one V2GCertificateChain installed.

Before (Preparations)
Configuration State: N/a
Memory State: RenewV2GChargingStationCertificate
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>GetInstalledCertificates</i> for certificateType V2GCertificateChain	

Tool validations
<p>* Step 1:</p> <p>Message: GetInstalledCertificateIdsResponse</p> <ul style="list-style-type: none"> - status must be <i>Accepted</i> - certificateHashDataChain must contain an entry with following values: <p><i>Note: Order does not matter.</i></p> <ul style="list-style-type: none"> - certificateType is V2GCertificateChain - certificateHashData uses the childCertificateHashData field
Post scenario validations: N/a

TC_M_16_CS: Retrieve certificates from Charging Station - MORootCertificate

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: <i>GetInstalledCertificates</i> from certificateType <i>MORootCertificate</i>
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>GetInstalledCertificates</i> for certificateType <i>MORootCertificate</i>	

Tool validations
N/a
Post scenario validations: N/a

TC_M_17_CS: Retrieve certificates from Charging Station - CSMSRootCertificate & ManufacturerRootCertificate

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)	<ul style="list-style-type: none"> - The Charging Station supports Security Profile 2 or 3. - The Charging Station supports signed firmware updates.

Before (Preparations)
Configuration State: N/a
Memory State: <i>CertificateInstalled</i> from certificateType CSMSRootCertificate <i>CertificateInstalled</i> from certificateType ManufacturerRootCertificate
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>GetInstalledCertificates</i> for certificateType CSMSRootCertificate AND ManufacturerRootCertificate	

Tool validations
N/a
Post scenario validations: N/a

TC_M_18_CS: Retrieve certificates from Charging Station - All certificateTypes

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)	<ul style="list-style-type: none"> - The Charging Station supports Security Profile 2 or 3. - The Charging Station supports signed firmware updates.

Before (Preparations)
Configuration State: N/a
Memory State: <i>CertificateInstalled</i> from certificateType CSMSRootCertificate <i>CertificateInstalled</i> from certificateType ManufacturerRootCertificate
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a GetInstalledCertificateIdsResponse	1. The Test System sends a GetInstalledCertificateIdsRequest With certificateType is omitted.

Tool validations
<p>* Step 2:</p> <p>Message: GetInstalledCertificateIdsResponse</p> <ul style="list-style-type: none"> - status must be <i>Accepted</i> - certificateHashDataChain must contain the following two entries with following values: <p>Note: Order does not matter.</p> <p>Entry 1:</p> <ul style="list-style-type: none"> - certificateHashDataChain[0].certificateType is <i>CSMSRootCertificate</i> - certificateHashDataChain[0].certificateHashData contains <i><HashData from configured new CSMS Root certificate></i> <p>Entry 2:</p> <ul style="list-style-type: none"> - certificateHashDataChain[1].certificateType is <i>ManufacturerRootCertificate</i> - certificateHashDataChain[1].certificateHashData contains <i><HashData from configured new Manufacturer Root certificate></i> <p>Post scenario validations: N/a </p>

TC_M_19_CS: Retrieve certificates from Charging Station - No matching certificate found

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: Test System 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 Test System sends a GetInstalledCertificateIdsRequest With certificateType is <i>MORootCertificate</i>

Tool validations
* Step 2: Message: GetInstalledCertificateIdsResponse - status must be <i>NotFound</i> - certificateHashDataChain must be omitted.
Post scenario validations: N/a

TC_M_20_CS: Delete a certificate from a Charging Station - Success

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 <i>GetInstalledCertificates</i> with certificateType CSMSRootCertificate	
3. The Charging Station responds with a DeleteCertificateResponse	2. The Test System sends a DeleteCertificateRequest with certificateHashData contains <i><Returned certificateHashData at step 1></i>
4. Execute Reusable State <i>GetInstalledCertificates</i> with certificateType CSMSRootCertificate	

Tool validations
<p>* Step 1:</p> <ul style="list-style-type: none"> - Certificate that is going to be deleted is present. <p>* Step 3:</p> <p>Message: DeleteCertificateResponse</p> <ul style="list-style-type: none"> - status must be <i>Accepted</i> <p>* Step 4:</p> <ul style="list-style-type: none"> - Certificate that should be deleted is not present anymore.
Post scenario validations: N/a

TC_M_22_CS: Delete a certificate from a Charging Station - No matching certificate found

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 <i>GetInstalledCertificates</i> with certificateType <i>CSMSRootCertificate</i> .	
3. The Charging Station responds with a DeleteCertificateResponse	2. The Test System sends a DeleteCertificateRequest with certificateHashData is <i><certificateHashData from unknown certificate></i>

Tool validations
* Step 3: Message: DeleteCertificateResponse - status must be <i>NotFound</i>
Post scenario validations: N/a

TC_M_23_CS: Delete a certificate from a Charging Station - Unable to delete the Charging Station Certificate

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)	<ul style="list-style-type: none"> - The Charging Station supports Security Profile 3. - A valid <i>CSMSRootCertificate</i> is installed on the Charging Station.

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): RenewChargingStationCertificate for certificateType <i>ChargingStationCertificate</i>

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>GetInstalledCertificates</i> with certificateType omitted.	
3. The Charging Station responds with a DeleteCertificateResponse	2. The Test System sends a DeleteCertificateRequest with certificateHashData is <certificateHashData from the generated ChargingStationCertificate at before.>

Tool validations
* Step 3: Message: DeleteCertificateResponse - status must be <i>NotFound</i> OR <i>Failed</i>
Post scenario validations: N/a

TC_M_24_CS: Get Charging Station Certificate status - Success

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 Test System responds with a GetCertificateStatusResponse with status <i>Accepted</i> ocspResult <i><OCSPResponse class as defined in IETF RFC 6960. DER encoded (as defined in IETF RFC 6960), and then base64 encoded.></i>
<u>Note:</u> Step 1/2 are repeated for the V2G Charging Station (leaf), the subCA1 and subCA2 certificates.	

Tool validations
N/a
Post scenario validations: N/a

TC_M_25_CS: Get Charging Station Certificate status - Rejected

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 <i>V2GRootCertificate</i> CertificateInstalled from certificateType <i>MORootCertificate</i> RenewV2GChargingStationCertificate
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
1. The Charging Station sends a GetCertificateStatusRequest	2. The Test System responds with a GetCertificateStatusResponse with status <i>Failed</i> ocspResult is omitted.

Tool validations
N/a
Post scenario validations: N/a

TC_M_26_CS: Certificate Installation EV - ISO15118-2 - Success

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 15118-2.

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 Test System 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 Test System responds with an AuthorizeResponse with status Accepted

Tool validations

* Step 1:

Message: **Get15118EVCertificateRequest**

- **action** must be *Install*
- **maximumContractCertificateChains** must be <omitted>
- **prioritizedEMAIDs** must be <omitted>

Post scenario validations:

N/a

TC_M_27_CS: Certificate Installation EV - Failed

Test case name	Certificate Installation EV - Failed
Test case Id	TC_M_27_CS
Use case Id(s)	M01
Requirement(s)	M01.FR.01, M01.FR.02
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 15118-2.

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 Test System responds with a Get15118EVCertificateResponse with status Failed exiResponse is omitted
	3. If an AuthorizeRequest is received, the testcase will FAIL and the Test System reports why it failed.

Tool validations

* Step 1:

Message: **Get15118EVCertificateRequest**

- **action** must be *Install*

If **iso15118SchemaVersion** contains ":15118:2:" then

- **maximumContractCertificateChains** must be *<omitted>*

- **prioritizedEMAIDs** must be *<omitted>*

Post scenario validations:

N/a

TC_M_28_CS: Certificate Update EV - Success

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 15118-2.

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 Test System responds with a Get15118EVCertificateResponse with status <i>Accepted</i> exiResponse <Raw <i>CertificateInstallationRes</i> response for the EV, Base64 encoded.>
3. The Charging Station sends an AuthorizeRequest	4. The Test System responds with an AuthorizeResponse with status <i>Accepted</i>

Tool validations

* Step 1:

Message: **Get15118EVCertificateRequest**

- **action** must be *Update*

If **iso15118SchemaVersion** contains ":15118:2:" then

- **maximumContractCertificateChains** must be <omitted>

- **prioritizedEMAIDs** must be <omitted>

Post scenario validations:

N/a

TC_M_29_CS: Certificate Update EV - Failed

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 15118-2.

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 Test System responds with a Get15118EVCertificateResponse with status Failed exiResponse is omitted.
	3. If an AuthorizeRequest is received, the testcase will FAIL and the Test System reports why it failed.

Tool validations

* Step 1:

Message: **Get15118EVCertificateRequest**

- **action** must be *Update*

If **iso15118SchemaVersion** contains ":15118:2:" then

- **maximumContractCertificateChains** must be *<omitted>*

- **prioritizedEMAIDs** must be *<omitted>*

Post scenario validations:

N/a

TC_M_100_CS: Certificate Installation EV - ISO 15118-20 - Success

Test case name	Certificate Installation EV - ISO 15118-20 - Success
Test case Id	TC_M_100_CS
Use case Id(s)	M01
Requirement(s)	M01.FR.01, M01.FR.04
System under test	Charging Station
Description	The EV initiates installing a new certificates. 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 and accepts multiple contracts.
Prerequisite(s)	- The Charging Station supports ISO 15118-20.

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 at most 5 certificates.

Memory State:

CertificateInstalled from certificateType *V2GRootCertificate*
CertificateInstalled from certificateType *MORootCertificate*

Reusable State(s):

State is *EVConnectedPreSession*

Main (Test scenario)

Charging Station	CSMS
<u>Note:</u> EV emulator accepts at most 5 contracts.	
1. The Charging Station sends a Get15118EVCertificateRequest	2. The Test System responds with a Get15118EVCertificateResponse with status <i>Accepted</i> exiResponse <Raw <i>CertificateInstallationRes</i> response for the EV, Base64 encoded.> remainingContracts <Configured <i>numberOfSupportedEvCertificates</i> > - 1
3. The Charging Station sends a Get15118EVCertificateRequest	4. The Test System responds with a Get15118EVCertificateResponse with status <i>Accepted</i> exiResponse <Raw <i>CertificateInstallationRes</i> response for the EV, Base64 encoded.> remainingContracts <Configured <i>numberOfSupportedEvCertificates</i> > - 2
5. The Charging Station sends a Get15118EVCertificateRequest	6. The Test System responds with a Get15118EVCertificateResponse with status <i>Accepted</i> exiResponse <Raw <i>CertificateInstallationRes</i> response for the EV, Base64 encoded.> remainingContracts <Configured <i>numberOfSupportedEvCertificates</i> > - 3

Tool validations
<p>* Step 1:</p> <p>Message: Get15118EVCertificateRequest</p> <ul style="list-style-type: none">- action must be <i>Install</i>- maximumContractCertificateChains must be 5 <p>* Step 3:</p> <p>Message: Get15118EVCertificateRequest</p> <ul style="list-style-type: none">- action must be <i>Install</i>- maximumContractCertificateChains must be 5 <p>* Step 5:</p> <p>Message: Get15118EVCertificateRequest</p> <ul style="list-style-type: none">- action must be <i>Update</i>- maximumContractCertificateChains must be 5
<p>Post scenario validations:</p> <p>N/a</p>

TC_M_101_CS: Install CA certificate - OEMRootCertificate

Test case name	Install CA certificate - OEMRootCertificate
Test case Id	TC_M_101_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 OEMRootCertificate.
Prerequisite(s)	- The Charging Station supports ISO15118-20

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

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>CertificateInstalled</i> for certificateType <i>OEMRootCertificate</i> (Root 2)	
<u>Note(s):</u> - When the Charging Station has the following configuration; AdditionalRootCertificateCheck implemented with value true , then a custom <i>OEMRootCertificate</i> should be used.	
2. Execute Reusable State <i>GetInstalledCertificates</i> for certificateType <i>OEMRootCertificate</i>	

Tool validations
N/a
Post scenario validations: N/a

N Diagnostics

TC_N_01_CS: Get Monitoring Report - with monitoringCriteria

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 , N02.FR.22
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 must have been configured by CSMS for Component Variable <Configured threshold monitor component variable>:

- LowerThreshold using value <Configured threshold monitor component variable LowerThreshold trigger value>
- UpperThreshold using value <Configured threshold monitor component variable UpperThreshold trigger value>
- Periodic using value <Configured Clock Aligned MeterValues Interval>

+ The following monitors must have been configured by CSMS for Component Variable <Configured numeric delta component variable>:

- Delta using value <Configured numeric delta component variable Delta numeric trigger value>
- PeriodicClockAligned using value <Configured Clock Aligned MeterValues Interval>

Memory State:

N/a

Reusable State(s):

N/a

Main (Test scenario)

Charging Station	CSMS
2. Charging Station responds with: GetMonitoringReportResponse	1. Test System sends GetMonitoringReportRequest with: - requestId = <Generated requestId> - monitoringCriteria = { <i>ThresholdMonitoring</i> }
3. Charging Station responds with: NotifyMonitoringReportRequest	4. Test System sends NotifyMonitoringReportResponse
Step 3 and 4 are repeated as often as needed to report all configuration variables.	
6. Charging Station responds with: GetMonitoringReportResponse	5. Test System sends GetMonitoringReportRequest with: - requestId = <Generated requestId> - monitoringCriteria = { <i>ThresholdMonitoring</i> , <i>DeltaMonitoring</i> }
7. Charging Station responds with: NotifyMonitoringReportRequest	8. Test System sends NotifyMonitoringReportResponse
Step 7 and 8 are repeated as often as needed to report all configuration variables.	
10. Charging Station responds with: GetMonitoringReportResponse	9. Test System sends GetMonitoringReportRequest with: - requestId = <Generated requestId> - monitoringCriteria = { <i>DeltaMonitoring</i> , <i>PeriodicMonitoring</i> }
11. Charging Station responds with: NotifyMonitoringReportRequest	12. Test System sends NotifyMonitoringReportResponse
Step 11 and 12 are repeated as often as needed to report all configuration variables.	

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*
- **monitor.variableMonitoring.eventNotificationType** = *<not omitted>*

While **tbc** = *true*

Message: **NotifyMonitoringReportRequest**

- **seqNo** is incremented by 1
- **monitor.variableMonitoring.type** = *UpperThreshold* or *LowerThreshold*
- **monitor.variableMonitoring.eventNotificationType** = *<not omitted>*

* 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*
- **monitor.variableMonitoring.eventNotificationType** = *<not omitted>*

While **tbc** = *true*

Message: **NotifyMonitoringReportRequest**

- **seqNo** is incremented by 1
- **monitor.variableMonitoring.type** = *UpperThreshold, LowerThreshold* or *Delta*
- **monitor.variableMonitoring.eventNotificationType** = *<not omitted>*

* 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*
- **monitor.variableMonitoring.eventNotificationType** = *<not omitted>*

While **tbc** = *true*

Message: **NotifyMonitoringReportRequest**

- **seqNo** is incremented by 1
- **monitor.variableMonitoring.type** = *Delta, Periodic* or *PeriodicClockAligned*
- **monitor.variableMonitoring.eventNotificationType** = *<not omitted>*

Post scenario validations:

N/A

TC_N_02_CS: Get Monitoring Report - with component/variable

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. Test System sends GetMonitoringReportRequest with: <ul style="list-style-type: none"> - 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. Test System sends NotifyMonitoringReportResponse
Step 3 and 4 are repeated as often as needed to report all configuration variables.	

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.

Tool validations	
While tbc = <i>true</i>	Message: NotifyMonitoringReportRequest <ul style="list-style-type: none">- seqNo is incremented by 1- monitor.variable = <i>"AvailabilityState"</i>- monitor.variableMonitoring.type = <i>Delta</i>- monitor.component_.name = <i>ChargingStation</i> or <i>EVSE</i>
Post scenario validations: <p>Check that a monitor for AvailabilityState of type <i>Delta</i> is reported for both ChargingStation and COnfigured EVSE. If other monitors are present on Configured EVSE, then they will also be reported.</p>	

TC_N_03_CS: Get Monitoring Report - with component criteria and component/variable

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 evseld>, 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. Test System sends GetMonitoringReportRequest with: <ul style="list-style-type: none"> - requestId = <Generated requestId1> - monitoringCriteria is <i>ThresholdMonitoring</i> - componentVariable[0].component.name = "ChargingStation" - componentVariable[0].variable.name = "AvailabilityState" - componentVariable[1].component.name = "EVSE" - componentVariable[1].component.evse.id = <Configured evseld> - componentVariable[1].variable.name = "AvailabilityState" <i>Note: requesting AvailabilityState from ChargingStation and Configured EVSE, but filtered to _ThresholdMonitoring._</i>
4. Charging Station responds with: GetMonitoringReportResponse	3. Test System sends GetMonitoringReportRequest with: <ul style="list-style-type: none"> - requestId = <Generated requestId2> - monitoringCriteria is <i>DeltaMonitoring</i> - componentVariable[0].component.name = "ChargingStation" - componentVariable[0].variable.name = "AvailabilityState" - componentVariable[1].component.name = "EVSE" - componentVariable[1].component.evse.id = <Configured evseld> - componentVariable[1].variable.name = "AvailabilityState" <i>Note: requesting AvailabilityState from ChargingStation and Configured EVSE, but filtered to _Delta._</i>
5. Charging Station responds with: NotifyMonitoringReportRequest	6. Test System sends NotifyMonitoringReportResponse
<i>Step 5 and 6 are repeated as often as needed to report all configuration variables.</i>	

Tool validations
<p>* Step 2:</p> <p>Message: GetMonitoringReportResponse</p> <ul style="list-style-type: none">- status = <i>EmptyResultSet</i>- statusInfo is absent or statusInfo.reasonCode = <i>"NotFound"</i> <p>* Step 4:</p> <p>Message: GetMonitoringReportResponse</p> <ul style="list-style-type: none">- status = <i>Accepted</i>- statusInfo is absent or statusInfo.reasonCode = <i>"NoError"</i> <p>* Step 5:</p> <p>Message: NotifyMonitoringReportRequest</p> <ul style="list-style-type: none">- requestId = <i><Generated requestId></i>- generatedAt = <i><timestamp at charging station></i>- seqNo = <i>0</i>- monitor.variableMonitoring.type = <i>Delta</i> <p>While tbc = <i>true</i></p> <p>Message: NotifyMonitoringReportRequest</p> <ul style="list-style-type: none">- seqNo is incremented by 1- monitor.variableMonitoring.type = <i>Delta</i>
<p>Post scenario validations:</p> <p>Check that nothing is reported for requestId = <i><Generated requestId1></i> and a monitor for AvailabilityState of type <i>Delta</i> is reported for both ChargingStation and EVSE #1 for requestId = <i><Generated requestId2></i>.</p>

TC_N_04_CS: Get Monitoring Report - for unknown component criteria

NOTE

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 <i>NotSupported</i> 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. Test System sends GetMonitoringReportRequest with: - requestId = <Generated requestId> - monitoringCriteria = { <i>DeltaMonitoring</i> , <Configured <i>Unsupported monitoringCriteria</i> > } - *componentVariable is absent

Tool validations
* Step 1 Message: GetMonitoringReportResponse - status = <i>NotSupported</i> - statusInfo is absent or statusInfo.reasonCode = " <i>UnsupportedParam</i> " or statusInfo.reasonCode = " <i>InvalidValue</i> "
Post scenario validations: N/A

TC_N_05_CS: Set Monitoring Base - success

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. Test System sends SetMonitoringBaseRequest with: - monitoringBase = <i>All</i>
4. Charging Station responds with: SetMonitoringBaseResponse	3. Test System sends SetMonitoringBaseRequest with: - monitoringBase = <i>FactoryDefault</i>
6. Charging Station responds with: SetMonitoringBaseResponse	5. Test System sends SetMonitoringBaseRequest with: - monitoringBase = <i>HardWiredOnly</i>

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

TC_N_06_CS: Set Monitoring Base - test removal custom monitors

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 <i>Delta</i> 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. <i>Note: this is a required variable for which a monitor can be expected to exist or it can be configured.</i>
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. Test System 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. Test System sends NotifyMonitoringReportResponse
6. Charging Station responds with: SetMonitoringBaseResponse	5. Test System sends SetMonitoringBaseRequest with: - monitoringBase = HardWiredOnly
Check that monitor AvailabilityState has been removed.	
8. Charging Station responds with: GetMonitoringReportResponse	7. Test System sends GetMonitoringReportRequest with: - requestId = <Generated requestId> - monitoringCriteria is absent - componentVariable[0].component.name = "ChargingStation" - componentVariable[0].variable.name = "AvailabilityState"

Tool validations
* Step 2 Message: GetMonitoringReportResponse - status = Accepted - statusInfo is absent or statusInfo.reasonCode = "NoError"

Tool validations
<p>* Step 3:</p> <p>Message: NotifyMonitoringReportRequest</p> <ul style="list-style-type: none">- requestId = <i><Generated requestId></i>- generatedAt = <i><timestamp at charging station></i>- seqNo = <i>0</i>- tbc is absent or tbc = <i>false</i>- monitor.variableMonitoring.type = <i>Delta</i>- monitor.component.name = <i>"ChargingStation"</i>- monitor.variable.name = <i>"AvailabilityState"</i>
<p>* Step 6</p> <p>Message: SetMonitoringBaseResponse</p> <ul style="list-style-type: none">- status = <i>Accepted</i>- statusInfo is absent or statusInfo.reasonCode = <i>"NoError"</i>
<p>* Step 8</p> <p>Message: GetMonitoringReportResponse</p> <ul style="list-style-type: none">- status = <i>EmptyResultSet</i>- statusInfo is absent or statusInfo.reasonCode = <i>"NotFound"</i>
<p>Post scenario validations:</p> <p>N/A</p>

TC_N_07_CS: Set Monitoring Base - for unknown base type

NOTE

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 <i>NotSupported</i> 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. Test System sends SetMonitoringBaseRequest with: - monitoringBase = <Configured unsupported_monitoringBase>

Tool validations
* Step 2 Message: SetMonitoringBaseResponse - status = <i>NotSupported</i> - statusInfo is absent or statusInfo.reasonCode = " <i>UnsupportedParam</i> " or statusInfo.reasonCode = " <i>InvalidValue</i> "
Post scenario validations: N/A

TC_N_08_CS: Set Variable Monitoring - one setMonitoringData element

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	<i>Install monitor</i> 1. Test System sends SetVariableMonitoringRequest with: - setMonitoringData.value = 1 - setMonitoringData.type = <i>Delta</i> - setMonitoringData.severity = 8 - setMonitoringData.component.name = "EVSE" - setMonitoringData.component.evse.id = <Configured evseld> - setMonitoringData.variable.name = "AvailabilityState"
4. Charging Station responds with: GetMonitoringReportResponse	<i>Verify monitor is installed</i> 3. Test System sends GetMonitoringReportRequest with: - requestId = <Generated requestId> - monitoringCriteria is absent - componentVariable[0].component.name = "EVSE" - componentVariable[0].component.evse.id = <Configured evseld> - componentVariable[0].variable.name = "AvailabilityState"
5. Charging Station responds with: NotifyMonitoringReportRequest	6. Test System sends NotifyMonitoringReportResponse

Tool validations
<p>* Step 2:</p> <p>Message: SetVariableMonitoringResponse with:</p> <pre>setMonitoringResult = { - status = Accepted - type = Delta - severity = 8 - component.name = "EVSE" - component.evse.id = <Configured evseld> - variable.name = "AvailabilityState" - statusInfo is absent or statusInfo.reasonCode = "NoError" }</pre>
<p>* Step 4:</p> <p>Message: GetMonitoringReportResponse</p> <pre>- status = Accepted - statusInfo is absent or statusInfo.reasonCode = "NoError"</pre>
<p>* Step 5:</p> <p>Message: NotifyMonitoringReportRequest</p> <pre>- requestId = <Generated requestId> - monitor.variableMonitoring.type = Delta - monitor.component.name = "EVSE" - monitor.component.evse.id = <Configured evseld> - monitor.variable.name = "AvailabilityState"</pre>
Post scenario validations:

TC_N_09_CS: Set Variable Monitoring - Multiple elements on different component and variable

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)
<p>Configuration State:</p> <p>This test case activates monitors on the following variables:</p> <ul style="list-style-type: none"> - Component "EVSE", evse "1", variable "AvailabilityState", monitor type <i>Delta</i> - Component "ChargingStation", variable "AvailabilityState", monitor type <i>Delta</i> <p>It assumes, that no monitor is active on these variables prior to the test.</p> <p><i>Note: these are required variables for which a monitor can be expected to exist or it can be configured.</i></p> <p><i>Note 2: Any other component/variable combination that supports monitoring could also be used for this test case.</i></p> <p>Memory State: N/a</p> <p>Reusable State(s): N/a</p>

Main (Test scenario)	
Charging Station	CSMS
<p>2. Charging Station responds with: SetVariableMonitoringResponse</p>	<p><i>Install monitors</i></p> <p>1. Test System sends SetVariableMonitoringRequest with:</p> <ul style="list-style-type: none"> - setMonitoringData[0].value = 1 - setMonitoringData[0].type = <i>Delta</i> - setMonitoringData[0].severity = <i><Configured severity></i> - setMonitoringData[0].component.name = "EVSE" - setMonitoringData[0].component.evse.id = <i><Configured evseld></i> - setMonitoringData[0].variable.name = "AvailabilityState" - setMonitoringData[1].value = 1 - setMonitoringData[1].type = <i>Delta</i> - setMonitoringData[1].severity = <i><Configured severity></i> - setMonitoringData[1].component.name = "ChargingStation" - setMonitoringData[1].variable.name = "AvailabilityState"
<p>4. Charging Station responds with: GetMonitoringReportResponse</p>	<p><i>Verify monitors are installed</i></p> <p>3. Test System sends GetMonitoringReportRequest with:</p> <ul style="list-style-type: none"> - requestId = <i><Generated requestId></i> - 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"

Main (Test scenario)	
5. Charging Station responds with: NotifyMonitoringReportRequest	6. Test System sends NotifyMonitoringReportResponse
Step 5 and 6 may be repeated if the result is not sent in one report message.	

Tool validations	
<p>* Step 2:</p> <p>Message: SetVariableMonitoringResponse with:</p> <p>setMonitoringResult[1]</p> <ul style="list-style-type: none"> - id = <id of new monitor> - status = <i>Accepted</i> - type = <i>Delta</i> - severity = 8 - component.name = "EVSE" - component.evse.id = <Configured evseId> - variable.name = "AvailabilityState" - statusInfo is absent or statusInfo.reasonCode = "NoError" <p>setMonitoringResult[2]</p> <ul style="list-style-type: none"> - id = <id of new monitor> - status = <i>Accepted</i> - type = <i>Delta</i> - severity = 8 - component.name = "ChargingStation" - variable.name = "AvailabilityState" - statusInfo is absent or statusInfo.reasonCode = "NoError" <p>* Step 4:</p> <p>Message: GetMonitoringReportResponse</p> <ul style="list-style-type: none"> - status = <i>Accepted</i> - statusInfo is absent or statusInfo.reasonCode = "NoError" * Step 5: <p>Message: NotifyMonitoringReportRequest</p> <ul style="list-style-type: none"> - requestId = <Generated requestId> - generatedAt = <timestamp at charging station> - seqNo = 0 	
while tb is true	<p>Expect NotifyMonitoringReportRequest</p> <ul style="list-style-type: none"> - seqNo is incremented by 1
<p>Post scenario validations:</p> <p>Verify the following monitors are reported in arbitrary order:</p> <p>monitor[1] = {</p> <ul style="list-style-type: none"> - variableMonitoring.type = <i>Delta</i> - variableMonitoring.severity = 8 - monitor.component.name = "EVSE" - monitor.component.evse.id = 1 - monitor.variable.name = "AvailabilityState" } <p>monitor[2] = {</p> <ul style="list-style-type: none"> - variableMonitoring.type = <i>Delta</i> - variableMonitoring.severity = 8 - monitor.component.name = "ChargingStation" - monitor.variable.name = "AvailabilityState" } 	

TC_N_10_CS: Set Variable Monitoring - Multiple monitors on the same component and variable

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.

<p>Before (Preparations)</p> <p>Configuration State: This test case activates two monitors on the following variable: - Component "EVSE", evse "1", variable "AvailabilityState", monitor type <i>Delta</i></p> <p><i>Note: it does not make any practical sense to install two <i>_Delta</i> monitors on same variable with different severity, because a <i>Delta</i> 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.</i> 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._</p> <p>Memory State: N/a</p> <p>Reusable State(s): N/a</p>

Main (Test scenario)	
Charging Station	CSMS
<p>2. Charging Station responds with: SetVariableMonitoringResponse</p>	<p><i>Install monitors</i></p> <p>1. Test System sends SetVariableMonitoringRequest with:</p> <ul style="list-style-type: none"> - setMonitoringData[0].value = 1 - setMonitoringData[0].type = <i>Delta</i> - 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 = <i>Delta</i> - setMonitoringData[1].severity = 7 - setMonitoringData[1].component.name = "EVSE" - setMonitoringData[1].component.evse.id = <Configured evseId> - setMonitoringData[1].variable.name = "AvailabilityState"

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 evseld>
- **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 evseld>
- **variable.name** = "AvailabilityState"
- **statusInfo** is absent or **statusInfo.reasonCode** = "NoError"
}

Post scenario validations:
N/A

TC_N_11_CS: Set Variable Monitoring - Unknown component

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	<p><i>Install monitors</i></p> <p>1. Test System sends SetVariableMonitoringRequest with:</p> <pre>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" }</pre>

Tool validations
<p>* Step 2:</p> <p>Message: SetVariableMonitoringResponse with (in arbitrary order):</p> <ul style="list-style-type: none">- id is absent- status = <i>UnknownComponent</i> or <i>Rejected</i>- type = <i>Delta</i>- severity = <i><Configured severity></i>- component.name = <i>"EVSE"</i>- component.evse.id = 99- variable.name = <i>"AvailabilityState"</i>- statusInfo is absent or statusInfo.reasonCode = <i>"UnknownEVSE"</i> or statusInfo = <i>"NotFound"</i>- id is absent- status = <i>UnknownComponent</i> (<i>UnknownVariable</i> will also be allowed, but is less accurate)- type = <i>UpperThreshold</i>- severity = <i><Configured severity></i>- component.name = <i>"NonExistent"</i>- variable.name = <i>"AvailabilityState"</i>- statusInfo is absent or statusInfo.reasonCode = <i>"NotFound"</i>
<p>Post scenario validations:</p> <p>N/A</p>

TC_N_12_CS: Set Variable Monitoring - Value out of range - Delta monitor

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 a numeric component variable exists which 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	<i>Install monitors</i> 1. Test System sends SetVariableMonitoringRequest with: - setMonitoringData[0].value = -1 - setMonitoringData[0].type = <i>Delta</i> - setMonitoringData[0].severity = <i><Configured severity></i> - setMonitoringData[0].component = <i><Configured numeric delta component></i> - setMonitoringData[0].variable = <i><Configured numeric delta component variable></i>

Tool validations
* Step 2: Message: SetVariableMonitoringResponse with (in arbitrary order): setMonitoringResult = { - id is absent - status = <i>Rejected</i> - type = <i>Delta</i> - severity = <Configured severity> - component = <Configured numeric delta component variable> - variable = <Configured numeric delta component variable> - statusInfo is absent or statusInfo.reasonCode = "ValueOutOfRange" or statusInfo.reasonCode = "ValuePositiveOnly" } Post scenario validations: N/A

TC_N_13_CS: Set Variable Monitoring - Value out of range - Threshold monitor

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.06
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 with maxLimit> component.variable exists and can be monitored and has variableCharacteristics.maxLimit < <Configured threshold monitor component variable with maxLimit exceeding maxLimit 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	<i>Install monitors</i> 1. Test System sends SetVariableMonitoringRequest with: - setMonitoringData[0].value = <Configured threshold monitor component variable with maxLimit exceeding maxLimit value> - setMonitoringData[0].type = UpperThreshold - setMonitoringData[0].severity = <Configured severity> - setMonitoringData[0].component = <Configured threshold monitor component variable with maxLimit> - setMonitoringData[0].variable = <Configured threshold monitor component variable with maxLimit>

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

TC_N_15_CS: Set Variable Monitoring - Duplicate Variable type/severity combination

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.

<p>Before (Preparations)</p> <p>Configuration State: This test case activates two monitors on the following variable: - Component "EVSE", evse "1", variable "AvailabilityState", monitor type <i>Delta</i> + Note: it does not make any practical sense to install two <i>_Delta</i> monitors on same variable with different severity, because a <i>Delta</i> 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._</p> <p>Memory State: N/a</p> <p>Reusable State(s): N/a</p>
--

Main (Test scenario)	
Charging Station	CSMS
<p>2. Charging Station responds with: SetVariableMonitoringResponse</p>	<p><i>Install monitors with same severity and of type <i>_Delta</i></i></p> <p>1. Test System sends SetVariableMonitoringRequest with:</p> <ul style="list-style-type: none"> - setMonitoringData[0].value = 1 - setMonitoringData[0].type = <i>Delta</i> - setMonitoringData[0].severity = <i><Configured severity></i> - setMonitoringData[0].component.name = "EVSE" - setMonitoringData[0].component.evse.id = <i><Configured evseId></i> - setMonitoringData[0].variable.name = "AvailabilityState" - setMonitoringData[1].value = 1 - setMonitoringData[1].type = <i>Delta</i> - setMonitoringData[1].severity = <i><Configured severity></i> - setMonitoringData[1].component.name = "EVSE" - setMonitoringData[1].component.evse.id = <i><Configured evseId></i> - setMonitoringData[1].variable.name = "AvailabilityState"

Tool validations
<p>* Step 2:</p> <p>Message: SetVariableMonitoringResponse with (in arbitrary order):</p> <p>setMonitoringResult[1] = {</p> <ul style="list-style-type: none">- id = <i><id of new monitor></i>- status = <i>Accepted</i>- type = <i>Delta</i>- severity = <i><Configured severity></i>- component.name = <i>"EVSE"</i>- component.evse.id = <i><Configured evseld></i>- variable.name = <i>"AvailabilityState"</i>- statusInfo is absent or statusInfo.reasonCode = <i>"NoError"</i> <p>}</p> <p>setMonitoringResult[2] = {</p> <ul style="list-style-type: none">- status = <i>Duplicate</i>- type = <i>Delta</i>- severity = <i><Configured severity></i>- component.name = <i>"EVSE"</i>- component.evse.id = <i><Configured evseld></i>- variable.name = <i>"AvailabilityState"</i>- statusInfo is absent or statusInfo.reasonCode = <i>"InvalidValue"</i> <p>}</p>
<p>Post scenario validations:</p> <p>N/A</p>

TC_N_16_CS: Set Monitoring Level - Success

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
<i>Set a monitoring level that suppresses the notification</i>	
2. Charging Station responds with: SetMonitoringLevelResponse	1. Test System sends: SetMonitoringLevelRequest with: severity = 7
4 Charging Station does NOT send NotifyEventRequest for configured EVSE	3. Plugin cable to configured EVSE to make it <i>_Occupied</i> 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.

TC_N_17_CS: Set Monitoring Level - Out of range

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. Test System sends: SetMonitoringLevelRequest with: severity = 10
4. Charging Station responds with: SetMonitoringLevelResponse	3. Test System sends: SetMonitoringLevelRequest with: severity = -1

Tool validations
<p>* Step 2:</p> <p>Message: SetMonitoringLevelResponse with:</p> <ul style="list-style-type: none"> - status = <i>Rejected</i> - statusInfo is absent or statusInfo.reasonCode = "ValueOutOfRange" or statusInfo.reasonCode = "ValueTooHigh"
<p>* Step 4:</p> <p>Message: SetMonitoringLevelResponse with:</p> <ul style="list-style-type: none"> - status = <i>Rejected</i> - statusInfo is absent or statusInfo.reasonCode = "ValueOutOfRange" or statusInfo.reasonCode = "ValueTooLow"
<p>Post scenario validations:</p> <p>N/A</p>

TC_N_18_CS: Clear Monitoring - Success

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. Test System sends: ClearVariableMonitoringRequest with: - id = { ID1, ID2 }
4. Charging Station responds with: GetMonitoringReportResponse	Verify monitors are cleared 3. Test System 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

TC_N_19_CS: Clear Monitoring - Not found

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 <i>NotFound</i> result.
Prerequisite(s)	Charging Station has implemented device model monitoring and MonitoringCtrlr::Enabled = true.

Before (Preparations)

Configuration State:

Test System Sends a GetMonitoringReportRequest, the CS then reports all existings 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. Test System sends: ClearVariableMonitoringRequest with: - <i>id monitor id from the Preparations</i>
--	--

Tool validations

* Step 2:

Message: **ClearVariableMonitoringResponse** with:

clearMonitoringResult:

- **status** = *NotFound*

- **id** = 123456

- **statusInfo** is absent or **statusInfo.reasonCode** = "*NotFound*"

Post scenario validations:

N/A

TC_N_20_CS: Alert Event - Threshold value exceeded

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 component variable *UpperThreshold* trigger value>

- **setMonitoringData[0].type** = *UpperThreshold*

- **setMonitoringData[0].severity** = 5

- **setMonitoringData[0].component** = <Configured threshold monitor component variable>

- **setMonitoringData[0].variable** = <Configured threshold monitor component variable>

Set MonitoringLevel to 8

Notes:

- Take a threshold that can easily be exceeded.

Reusable State(s): N/a

Main (Test scenario)

Charging Station	CSMS
1. Execute Reusable State <i>EnergyTransferStarted</i> or manually trigger the monitor. <i>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.</i>	
2. Charging Station sends a NotifyEventRequest with: - Power exceeding upper threshold	3. Test System responds with a NotifyEventResponse
5. Charging Station responds with a SetVariableMonitoringResponse with: - status <i>Accepted</i>	4. Test System sends a SetVariableMonitoringRequest with: - type <i>LowerThreshold</i> - component <Configured threshold monitor component variable> - variable <Configured threshold monitor component variable> - value <Configured threshold monitor component variable <i>LowerThreshold</i> trigger value> <i>Notes:</i> - Take a threshold that won't be exceeded.
6. Execute Reusable State <i>StopAuthorized</i> or manually trigger the second monitor. <i>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.</i>	
7. Charging Station sends: NotifyEventRequest for 2 events: - Returning below upper threshold (<i>cleared</i>) - Dropping below lower threshold	8. Test System responds: NotifyEventResponse

Main (Test scenario)

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** = <Configured threshold monitor component variable>
 - **variable.name** = <Configured threshold monitor component variable>

Other **eventData** elements can be ignored.

* 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** = <Configured threshold monitor component variable>
 - **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** = <Configured threshold monitor component variable>
- **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

TC_N_21_CS: Alert Event - Caused by hardwired trigger

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 <i>HardWiredNotification</i> .
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 <i>NotifyEventRequest</i> .

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. Test System responds: NotifyEventResponse

Tool validations
<p>* Step 1: Message: NotifyEventRequest with:</p> <ul style="list-style-type: none"> - generatedAt = <time of generation at Charging Station> - seqNo = 0 <p>and an eventData element with:</p> <ul style="list-style-type: none"> - eventNotificationType = <i>HardWiredNotification</i> <p>Other eventData elements are not relevant for this test.</p>
Post scenario validations: N/A

TC_N_22_CS: Offline Notification - OfflineMonitoringEventQueuingSeverity set equal or lower

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 <code>OfflineMonitoringEventQueuingSeverity</code> .
Prerequisite(s)	Charging Station is online at start of test for configuration. CS has implemented device model monitoring and <code>MonitoringCtrlr::Enabled = true</code> .

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._
<u>Note(s)</u> : The tool will now wait for <Configured Transaction Duration> seconds	
<u>Manual Action</u> : Bring Charging Station back online.	
3. Charging Station sends NotifyEventRequest	4. Test System 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

TC_N_23_CS: Offline Notification - OfflineMonitoringEventQueuingSeverity set higher than severityLevel of the monitor

Test case name	Offline Notification - OfflineMonitoringEventQueuingSeverity set higher than severityLevel of the monitor
Test case Id	TC_N_23_CS
Use case Id(s)	N07
Requirement(s)	N07.FR.04
System under test	Charging Station
Description	Charging Station does not queue event notifications when offline.
Purpose	To test that Charging Station does not queue event notifications with a severity higher than OfflineMonitoringEventQueuingSeverity.
Prerequisite(s)	Charging Station is online at start of test for configuration. CS has implemented device model monitoring and MonitoringCtrlr::Enabled = true.

Before (Preparations)
Configuration State: SetConfiguration with: - component.name = "MonitoringCtrlr" - variable.name = "OfflineQueuingSeverity" - attributeValue = <Configured Severity>
Memory State: Charging Station has custom or predefined monitors on variable AvailabilityState of Configured EVSE and Configured ConnectorId with severity = <Configured severity> + 1
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
<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 Test System 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 Test System 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 Test System responds with a TransactionEventResponse
7. The Charging Station sends a TransactionEventRequest	8. The Test System responds with a TransactionEventResponse
<u>Note(s):</u> The CS shall not send a NotifyEventRequest for AvailabilityState of EVSE and Connector. A StatusNotification may still be received.	

Tool validations
<p>* Step 1: (Optional:)</p> <p>Message: StatusNotificationRequest</p> <ul style="list-style-type: none"> - evseld <configured evseld> - connectorId <configured connectorId> - connectorStatus must be <i>Occupied</i> <p>(Required, but can be combined into one NotifyEventRequest:)</p> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none"> - eventData[0].trigger must be <i>Delta</i> - eventData[0].actualValue must be <i>Occupied</i> - eventData[0].component.name must be <i>Connector</i> - eventData[0].component.evse.id must be <i>Configured EVSE</i> - eventData[0].component.evse.connectorId must be <i>Configured ConnectorId</i> - eventData[0].variable.name must be <i>AvailabilityState</i> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none"> - eventData[0].trigger must be <i>Delta</i> - eventData[0].actualValue must be <i>Occupied</i> - eventData[0].component.name must be <i>EVSE</i> - eventData[0].component.evse.id must be <i>Configured EVSE</i> - eventData[0].variable.name must be <i>AvailabilityState</i>
<p>* Step 3:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - triggerReason must be <i>CablePluggedIn</i> - transactionInfo.chargingState must be <i>EVConnected</i>
<p>* Step 5:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - triggerReason must be <i>EVCommunicationLost</i> - transactionInfo.chargingState must be <i>Idle</i>
<p>* Step 7:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - triggerReason must be <i>CablePluggedIn</i> - transactionInfo.chargingState must be <i>EVConnected</i>
<p>Post scenario validations:</p> <p>N/A</p>

TC_N_24_CS: Set Variable Monitoring - Periodic event

Test case name	Set Variable Monitoring - Periodic event
Test case Id	TC_N_24_CS
Use case Id(s)	N04, N08
Requirement(s)	N04.FR.01, N04.FR.08, N08.FR.05 and N08.FR.06
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:

Set MonitoringLevel to 8

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. Test System sends SetVariableMonitoringRequest with: <ul style="list-style-type: none"> - setMonitoringData[0].value = <Configured Clock Aligned MeterValues Interval> - setMonitoringData[0].type = Periodic - setMonitoringData[0].severity = 5 - setMonitoringData[0].component = <Configured threshold monitor component variable> - setMonitoringData[0].variable = <Configured threshold monitor component variable>
3. Charging Station sends a NotifyEventRequest	4. Test System responds with a NotifyEventResponse
<u>Note(s)</u> : Step 3 and 4 will repeat every <Configured Clock Aligned MeterValues Interval> seconds	

Tool validations

* Step 2:

Message: **SetVariableMonitoringResponse** with:**setMonitoringResult[0].status** = Accepted**setMonitoringResult[0].type** = Periodic**setMonitoringResult[0].severity** = 5**setMonitoringResult[0].component** = <Configured threshold monitor component variable>**setMonitoringResult[0].variable** = <Configured threshold monitor component variable>**setMonitoringResult[0].attributeStatusInfo** is absent or **attributeStatusInfo.reasonCode** = "NoError"

* Step 3:

Message: **NotifyEventRequest** every <Configured Clock Aligned MeterValues Interval> seconds with:with an **eventData** element with:- **trigger** = Periodic- **component** = <Configured threshold monitor component variable>- **variable** = <Configured threshold monitor component variable>

Post scenario validations:

N/A

TC_N_25_CS: Retrieve Log Information - Diagnostics Log - Success

Test case name	Retrieve Log Information - Diagnostics Log - Success
Test case Id	TC_N_25_CS
Use case Id(s)	N01
Requirement(s)	N01.FR.01, N01.FR.02, N01.FR.04, N01.FR.07, N01.FR.08, N01.FR.09, N01.FR.13
System under test	Charging Station
Description	This test case covers the functionality of getting log information from a Charging Station. The CSMS can request a Charging Station to upload a file with log information to a given location (URL). The format of this log file is not prescribed. The Charging Station successfully uploads a log file and gives information about the status of the upload by sending status notifications to the CSMS.
Purpose	To verify if the Charging station is able to successfully upload a log as described at the OCPP specification.
Prerequisite(s)	<ul style="list-style-type: none"> - Charging Station has log information available. - A diagnostics logging server has been setup supporting one of the file transfer protocols supported by the Charging Station (This is configured at the configuration variable FileTransferProtocols).

Before (Preparations)

Configuration State:

N/a

Memory State:

N/a

Reusable State:

N/a

Main (Test scenario)

Charging Station	CSMS
2. The Charging Station responds with a GetLogResponse	1. The Test System sends a GetLogRequest with logType <i>DiagnosticsLog</i>
<u>Note(s):</u> - <i>Charging Station is uploading log file</i>	
3. The Charging Station sends a LogStatusNotificationRequest	4. The Test System responds with a LogStatusNotificationResponse
<u>Note(s):</u> - <i>Log file is uploaded</i>	
5. The Charging Station sends a LogStatusNotificationRequest	6. The Test System 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

TC_N_26_CS: Retrieve Log Information - Diagnostics Log - Upload failed

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.

Reusable State:

N/a

Main (Test scenario)

Charging Station	CSMS
2. The Charging Station responds with a GetLogResponse	1. The Test System sends a GetLogRequest with - logType <i>DiagnosticsLog</i> - retries 3 - retryInterval <Configured retryInterval> - log.remoteLocation <Configured log location with a nonexistent path>
3. The Charging Station sends a LogStatusNotificationRequest	4. The Test System responds with a LogStatusNotificationResponse
5. The Charging Station sends a LogStatusNotificationRequest	6. The Test System responds with a LogStatusNotificationResponse

Note(s):

- Step 3-4, 5-6 and 3-6 may repeat multiple times depending on Charging Station's implementation.
- The Test System waits at least (3 * <Configured retryInterval>), before ending the testcase.

Tool validations
<p>* Step 2:</p> <p>Message GetLogResponse</p> <ul style="list-style-type: none">- status <i>Accepted</i> <p>* Step 3:</p> <p>Must be sent exactly 1 or 4 times</p> <p>Message LogStatusNotificationRequest</p> <ul style="list-style-type: none">- status <i>Uploading</i>- requestId <i>Same Id as the GetLogRequest</i> <p>* Step 5:</p> <p>Must be sent exactly 1 or 4 times</p> <p>Message LogStatusNotificationRequest</p> <ul style="list-style-type: none">- status <i>UploadFailure</i>- requestId <i>Same Id as the GetLogRequest</i> <p>OR Message LogStatusNotificationRequest</p> <ul style="list-style-type: none">- status <i>BadMessage</i>- requestId <i>Same Id as the GetLogRequest</i> <p>OR Message LogStatusNotificationRequest</p> <ul style="list-style-type: none">- status <i>PermissionDenied</i>- requestId <i>Same Id as the GetLogRequest</i> <p>OR Message LogStatusNotificationRequest</p> <ul style="list-style-type: none">- status <i>NotSupportedOperation</i>- requestId <i>Same Id as the GetLogRequest</i> <p>* The time between the first LogStatusNotificationRequest <i>Uploading</i> and the last LogStatusNotificationRequest <i>UploadFailure/BadMessage/PermissionDenied/NotSupportedOperation</i> equals $(3 * \text{<Configured retryInterval>})$</p>
<p>Post scenario validations:</p> <ul style="list-style-type: none">- N/a

TC_N_27_CS: Get Customer Information - Accepted + data

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)	<ul style="list-style-type: none"> - 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)

Reusable State:

State is *Authorized* (local)

State is *ParkingBayUnoccupied*

Main (Test scenario)

Charging Station	CSMS
2. The Charging Station responds with a CustomerInformationResponse	1. The Test System sends a CustomerInformationRequest with - report <i>true</i> - idToken <Configured valid idToken fields>
3. The Charging Station sends a NotifyCustomerInformationRequest	4. The Test System responds with a NotifyCustomerInformationResponse .
Note(s): - If <i>tbv</i> is <i>True</i> 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

TC_N_28_CS: Get Customer Information - Accepted + no data

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>
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a CustomerInformationResponse	1. The Test System sends a CustomerInformationRequest with - report true - idToken <Configured valid idToken fields>
3. The Charging Station sends a NotifyCustomerInformationRequest	4. The Test System responds with a NotifyCustomerInformationResponse .

Tool validations
* Step 2: Message CustomerInformationResponse - status Accepted
* Step 3: Message NotifyCustomerInformationRequest - tbv Not true
Post scenario validations: - A message is sent indicating that no data is found

TC_N_29_CS: Get Customer Information - Not Accepted

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a CustomerInformationResponse	1. The Test System sends a CustomerInformationRequest

Tool validations
* Step 2: Message CustomerInformationResponse - status <i>Invalid</i>
Post scenario validations: - N/a

TC_N_30_CS: Clear Customer Information - Clear and report + data

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)	<ul style="list-style-type: none"> - 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)

Reusable State:

State is *Authorized* (local)

State is *ParkingBayUnoccupied*

Main (Test scenario)

Charging Station	CSMS
2. The Charging Station responds with a CustomerInformationResponse	1. The Test System sends a CustomerInformationRequest with <ul style="list-style-type: none"> - report <i>true</i> AND - clear <i>true</i> AND - idToken <Configured valid idToken fields>
3. The Charging Station sends a NotifyCustomerInformationRequest	4. The Test System responds with a NotifyCustomerInformationResponse .
<u>Note(s):</u> - If tbc is <i>True</i> at Step 3 then step 3 and 4 will be repeated	
6. The Charging Station responds with a CustomerInformationResponse	5. The Test System sends a CustomerInformationRequest with <ul style="list-style-type: none"> - report <i>true</i> AND - idToken <Configured valid idToken fields>
7. The Charging Station sends a NotifyCustomerInformationRequest	8. The Test System responds with a NotifyCustomerInformationResponse .
<u>Note(s):</u> - Step is optional and only expected when status is <i>Accepted</i> at Step 6	

Tool validations
<div>* Step 2: Message CustomerInformationResponse - status <i>Accepted</i></div> <div>* Step 3: Message NotifyCustomerInformationRequest - data <i>Not empty</i></div> <div>* Step 8: Message NotifyCustomerInformationRequest - tbc <i>Not true</i></div>
<div>Post scenario validations: - All report parts have been received</div>

TC_N_31_CS: Clear Customer Information - Clear and report + no data

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a CustomerInformationResponse	1. The Test System sends a CustomerInformationRequest with - report <i>true</i> AND - clear <i>true</i> AND - idToken <Configured valid idToken fields>
3. The Charging Station sends a NotifyCustomerInformationRequest	4. The Test System responds with a NotifyCustomerInformationResponse .

Tool validations
* Step 2: Message CustomerInformationResponse - status <i>Accepted</i>
Post scenario validations: - A message is send indicating that no data is found

TC_N_32_CS: Clear Customer Information - Clear and no report

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a CustomerInformationResponse	1. The Test System sends a CustomerInformationRequest with - report <i>false</i> AND - clear <i>true</i> AND - idToken <Configured valid idToken fields>
3. The Charging Station sends a NotifyCustomerInformationRequest	4. The Test System responds with a NotifyCustomerInformationResponse .

Tool validations
* Step 2: Message CustomerInformationResponse - status <i>Accepted</i>
Post scenario validations: - A message is send indicating that the data is cleared

TC_N_62_CS: Clear Customer Information - Clear and report - customerIdentifier

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.
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a CustomerInformationResponse	1. The Test System sends a CustomerInformationRequest with - report <i>true</i> AND - clear <i>true</i> AND - customerIdentifier <Configured customerIdentifier>
3. The Charging Station sends a NotifyCustomerInformationRequest	4. The Test System responds with a NotifyCustomerInformationResponse
<u>Note(s):</u> - If tbc is <i>True</i> at Step 3 then step 3 and 4 will be repeated	
6. The Charging Station responds with a CustomerInformationResponse	5. The Test System sends a CustomerInformationRequest with - report <i>true</i> AND - clear <i>false</i> AND - customerIdentifier <Configured customerIdentifier>
7. The Charging Station sends a NotifyCustomerInformationRequest	8. The Test System responds with a NotifyCustomerInformationResponse
<u>Note(s):</u> - If tbc is <i>True</i> at Step 7 then step 7 and 8 will be repeated	

Tool validations
<p>* Step 2: Message CustomerInformationResponse - status <i>Accepted</i></p> <p>* Step 3: Message NotifyCustomerInformationRequest - data <i>Not empty</i></p> <p>* Step 6: Message CustomerInformationResponse - status <i>Accepted</i></p> <p>* Step 7: Message NotifyCustomerInformationRequest - data <i>empty</i></p>
<p>Post scenario validations: - All report parts have been received</p>

TC_N_63_CS: Clear Customer Information - Clear and report - customerCertificate

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	<p>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.</p> <p>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.</p>
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)
<p>Configuration State: - AuthCtrlr.Enabled is <i>true</i> - AuthCtrlr.DisableRemoteAuthorization is <i>false</i> - ISO15118Ctrlr.CentralContractValidationAllowed is <i>false</i> - ISO15118Ctrlr.V2GCertificateInstallationEnabled is <i>true</i> - ISO15118Ctrlr.ContractCertificateInstallationEnabled is <i>true</i> - ISO15118Ctrlr.PnCEnabled is <i>true</i></p>
<p>Memory State: RenewV2GChargingStationCertificate (If none are present, when checking with GetInstalledCertificateIds.certificateType = V2GCertificateChain)</p>
<p>Reusable State: Execute Reusable State EVConnectedPreSession Execute Reusable State Authorized15118 (PnC) Execute Reusable State EnergyTransferStarted</p>

Main (Test scenario)	
Charging Station	CSMS
<u>Manual action:</u> EV ends the charging session. <u>Note:</u> The Charging Station receives a SessionStopReq(Terminate) message from the EV to finish the transaction.	

Main (Test scenario)	
2. The Charging Station responds with a CustomerInformationResponse	1. The Test System sends a CustomerInformationRequest with - report <i>true</i> AND - clear <i>true</i> AND - customerCertificate <i>certificate hash data of contract certificate</i>
3. The Charging Station sends a NotifyCustomerInformationRequest	4. The Test System responds with a NotifyCustomerInformationResponse
<u>Note(s):</u> - If tbc is <i>True</i> at Step 3 then step 3 and 4 will be repeated	
6. The Charging Station responds with a CustomerInformationResponse	5. The Test System sends a CustomerInformationRequest with - report <i>true</i> AND - clear <i>false</i> AND - customerCertificate <i>certificate hash data of contract certificate</i>
7. The Charging Station sends a NotifyCustomerInformationRequest	8. The Test System responds with a NotifyCustomerInformationResponse

Tool validations
* Step 2: Message CustomerInformationResponse - status <i>Accepted</i> * Step 3: Message NotifyCustomerInformationRequest - data <i>empty</i> or <i>Not empty</i> if a customer certificate exists * Step 6: Message CustomerInformationResponse - status <i>Accepted</i> * Step 7: Message NotifyCustomerInformationRequest - data <i>empty</i>
Post scenario validations: - All report parts have been received

TC_N_33_CS: Clear Customer Information - Invalid

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a CustomerInformationResponse	1. The Test System sends a CustomerInformationRequest

Tool validations
* Step 2: Message CustomerInformationResponse - status <i>Invalid</i>
Post scenario validations: - N/a

TC_N_34_CS: Retrieve Log Information - Rejected

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a GetLogResponse	1. The Test System sends a GetLogRequest with logType <Configured logType>

Tool validations
* Step 2: Message GetLogResponse - status <i>Rejected</i>
Post scenario validations: - N/a

TC_N_35_CS: Retrieve Log Information - Security Log - Success

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.
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a GetLogResponse	1. The Test System sends a GetLogRequest with logType <i>SecurityLog</i>
Note(s): - Charging Station is uploading log file	
3. The Charging Station sends a LogStatusNotificationRequest	4. The Test System responds with a LogStatusNotificationResponse .
Note(s): - Log file is uploaded	
5. The Charging Station sends a LogStatusNotificationRequest	6. The Test System responds with a LogStatusNotificationResponse .

Tool validations
<p>* Step 2: Message GetLogResponse - status <i>Accepted</i></p> <p>* Step 3: Message LogStatusNotificationRequest - status <i>Uploading</i> - requestId <i>Same Id as the GetLogRequest</i></p> <p>* Step 5: Message LogStatusNotificationRequest - status <i>Uploaded</i> - requestId <i>Same Id as the GetLogRequest</i></p>
Post scenario validations: - N/a

TC_N_36_CS: Retrieve Log Information - Second Request

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>.

Reusable State:

N/a

Main (Test scenario)

Charging Station	CSMS
2. The Charging Station responds with a GetLogResponse	1. The Test System sends a GetLogRequest with logType <Configured logType> and requestId <#1>
<u>Note(s):</u> - Charging Station is uploading log file	
3. The Charging Station sends a LogStatusNotificationRequest	4. The Test System responds with a LogStatusNotificationResponse .
<u>Note(s):</u> - Charging Station cancels uploading the first log file	
6. The Charging Station responds with a GetLogResponse	5. The Test System sends a GetLogRequest with logType <Configured logType> and requestId <#2>
Step 7 is allowed to occur before step 6 7. The Charging Station sends a LogStatusNotificationRequest	8. The Test System responds with a LogStatusNotificationResponse .
<u>Note(s):</u> - Charging Station is uploading log file	
9. The Charging Station sends a LogStatusNotificationRequest	10. The Test System responds with a LogStatusNotificationResponse .
<u>Note(s):</u> - Log file is uploaded	
11. The Charging Station sends a LogStatusNotificationRequest	12. The Test System responds with a LogStatusNotificationResponse .

Tool validations
<p>* Step 2:</p> <p>Message GetLogResponse</p> <ul style="list-style-type: none"> - status <i>Accepted</i> <p>* Step 3:</p> <p>Message LogStatusNotificationRequest</p> <ul style="list-style-type: none"> - status <i>Uploading</i> - requestId <#1>
<p><i>Step 7 is allowed to occur before step 6</i></p> <p>* Step 6:</p> <p>Message GetLogResponse</p> <ul style="list-style-type: none"> - status <i>AcceptedCanceled</i> <p>* Step 7:</p> <p>Message LogStatusNotificationRequest</p> <ul style="list-style-type: none"> - status <i>AcceptedCanceled</i> - requestId <#1>
<p>* Step 9:</p> <p>Message LogStatusNotificationRequest</p> <ul style="list-style-type: none"> - status <i>Uploading</i> - requestId <#2> <p>* Step 11:</p> <p>Message LogStatusNotificationRequest</p> <ul style="list-style-type: none"> - status <i>Uploaded</i> - requestId <#2>
<p>Post scenario validations:</p> <ul style="list-style-type: none"> - N/a

TC_N_37_CS: Set Variable Monitoring - Unknown Variable

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetVariableMonitoringResponse	1. The Test System sends a SetVariableMonitoringRequest with setMonitoringData.type <i>Delta</i> setMonitoringData.variable.name <i>unknownVariable</i> setMonitoringData.component.name <i>EVSE</i>

Tool validations
<p>* Step 2:</p> <p>Message SetVariableMonitoringResponse</p> <ul style="list-style-type: none"> - setMonitoringResult[0].status <i>UnknownVariable</i> - setMonitoringResult[0].type <i>Delta</i> - setMonitoringResult[0].severity <i><Configured severity></i> - setMonitoringResult[0].component.name <i>EVSE</i> - setMonitoringResult[0].variable.name <i>unkownVariable</i>
Post scenario validations:
- N/a

TC_N_38_CS: Set Variable Monitoring - Not supported MonitorType

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)	<ul style="list-style-type: none"> - Charging Station supports Monitoring. - Charging station does not support one or more variableMonitoringTypes.

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetVariableMonitoringResponse	1. The Test System sends a SetVariableMonitoringRequest with setVariableData *setMonitoringData.type <i>UpperThreshold</i> setMonitoringData.variable.name <i>AvailabilityState</i> setMonitoringData.component.name <i>EVSE</i>

Tool validations
* Step 2: Message SetVariableMonitoringResponse - setMonitoringResult[0].status <i>UnsupportedMonitorType</i> or Rejected - setMonitoringResult[0].type <i>UpperThreshold</i> - setMonitoringResult[0].component.name <i>EVSE</i> - setMonitoringResult[0].variable.name <i>AvailabilityState</i>
Post scenario validations: - N/a

TC_N_39_CS: Set Variable Monitoring - Component/Variable combination does NOT correspond

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 This test case assumes the Charging Station has a non-numeric Component Variable <Configured non-numeric delta component variable> and numeric Component Variable <Configured threshold monitor component variable>.

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetVariableMonitoringResponse	1. The Test System sends a SetVariableMonitoringRequest with setMonitoringData[0].type <i>Delta</i> setMonitoringData[0].value 1 setMonitoringData[0].component = <Configured non-numeric delta component variable> setMonitoringData[0].variable = <Configured non-numeric delta component variable>
4. The Charging Station responds with a SetVariableMonitoringResponse	3. The Test System sends a SetVariableMonitoringRequest with setMonitoringData[0].id <SetVariableMonitoringResponse.setMonitoringResult[0].id of step 2> setMonitoringData[0].type <i>Delta</i> setMonitoringData[0].value 1 setMonitoringData[0].component = <Configured numeric delta component variable> setMonitoringData[0].variable = <Configured numeric delta component variable>
6. The Charging Station responds with a GetMonitoringReportResponse	5. The Test System sends a GetMonitoringReportRequest with - requestId <Generated requestId>
7. The Charging Station sends a NotifyMonitoringReportRequest	8. The Test System responds with a NotifyMonitoringReportResponse .
Note(s): - If tbc is True at Step 7 then step 7 and 8 will be repeated	

Tool validations
<p>* Step 2:</p> <p>Message SetVariableMonitoringResponse</p> <ul style="list-style-type: none">- setMonitoringResult[0].id <i><not omitted></i>- setMonitoringResult[0].status <i>Accepted</i>- setMonitoringResult[0].type <i>Delta</i>- setMonitoringResult[0].component = <i><Configured non-numeric delta component variable></i>- setMonitoringResult[0].variable = <i><Configured non-numeric delta component variable></i> <p>* Step 4:</p> <p>Message SetVariableMonitoringResponse</p> <ul style="list-style-type: none">- setMonitoringResult[0].status <i>Rejected</i>- setMonitoringResult[0].type <i>Delta</i>- setMonitoringResult[0].component = <i><Configured numeric delta component variable></i>- setMonitoringResult[0].variable = <i><Configured numeric delta component variable></i> <p>* Step 6:</p> <p>Message GetMonitoringReportResponse</p> <ul style="list-style-type: none">- status <i>Accepted</i> <p>* Step 7:</p> <p>Message NotifyMonitoringReportRequest</p> <p>Must contain a monitor with</p> <ul style="list-style-type: none">- monitor[0].component = <i><Configured non-numeric delta component variable></i>- monitor[0].variable = <i><Configured non-numeric delta component variable></i>- monitor[0].variableMonitoring[0].id = <i><SetVariableMonitoringResponse.setMonitoringResult[0].id of step 2></i>- monitor[0].variableMonitoring[0].value = <i>1</i>- monitor[0].variableMonitoring[0].type = <i>Delta</i> <p>Post scenario validations:</p> <ul style="list-style-type: none">- All report parts have been received

TC_N_40_CS: Set Variable Monitoring - Replace Variable Monitor

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

Reusable State:

N/a

Main (Test scenario)

Charging Station	CSMS
2. The Charging Station responds with a SetVariableMonitoringResponse	1. The Test System sends a SetVariableMonitoringRequest with setMonitoringData.id <Generated variableMonitoringId> AND setMonitoringData.type <i>Delta</i> setMonitoringData.severity 4
4. The Charging Station responds with a GetMonitoringReportResponse	3. The Test System sends a GetMonitoringReportRequest with - requestId <Generated requestId> - componentVariable.component.name <i>EVSE</i> - componentVariable.component.evse.id <i>evseId</i> - componentVariable.variable.name <i>AvailabilityState</i> - monitoringCriteria <i>DeltaMonitoring_</i>
5. The Charging Station sends a NotifyMonitoringReportRequest	6. The Test System 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

TC_N_41_CS: Set Variable Monitoring - Return to FactoryDefault

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 - A preconfigured monitor exists with <i>id</i> <Configured preconfigured monitor id>.

Before (Preparations)
Configuration State: N/a
Memory State: MonitoringBase has been set to <i>FactoryDefault</i>
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with GetMonitoringReportResponse	1. The Test System sends GetMonitoringReportRequest with requestId = <Generated requestId>
3. The Charging Station sends NotifyMonitoringReportRequest	4. The Test System responds with NotifyMonitoringReportResponse
<u>Note(s):</u> - If NotifyMonitoringReportRequest.tbc is True in Step 3 then step 3 and 4 will be repeated	
Search NotifyMonitoringReportRequest.monitoringReportData of step 3 to get a monitoringReportData.monitor WHERE monitor.variableMonitoring.id is <Configured preconfigured monitor id> AS <preconfiguredMonitor>	
6. The Charging Station responds with a SetVariableMonitoringResponse	5. The Test System sends a SetVariableMonitoringRequest with setMonitoringData.id <preconfiguredMonitor.variableMonitoring.id> setMonitoringData.transaction <preconfiguredMonitor.variableMonitoring.transaction> setMonitoringData.value <preconfiguredMonitor.variableMonitoring.value> setMonitoringData.type <preconfiguredMonitor.variableMonitoring.type> setMonitoringData.severity IF <preconfiguredMonitor.variableMonitoring.severity> < 9 THEN <preconfiguredMonitor.variableMonitoring.severity> + 1 ELSE 5 END IF setMonitoringData.component <preconfiguredMonitor.component> setMonitoringData.variable <preconfiguredMonitor.variable>

Main (Test scenario)	
8. The Charging Station responds with a GetMonitoringReportResponse	7. The Test System sends a GetMonitoringReportRequest with - requestId <Generated requestId> - id <preconfiguredMonitor.variableMonitoring.id> - componentVariable.component <preconfiguredMonitor.component> - componentVariable.variable <preconfiguredMonitor.variable>
9. The Charging Station sends a NotifyMonitoringReportRequest	10. The Test System responds with a NotifyMonitoringReportResponse .
12. The Charging Station responds with a SetMonitoringBaseResponse with - status <i>Accepted</i>	11. The Test System sends a SetMonitoringBaseRequest with - monitoringBase <i>FactoryDefault</i>
14. The Charging Station responds with a GetMonitoringReportResponse	13. The Test System sends a GetMonitoringReportRequest with - requestId <Generated requestId> - id <preconfiguredMonitor.variableMonitoring.id> - componentVariable.component <preconfiguredMonitor.component> - componentVariable.variable <preconfiguredMonitor.variable>
15. The Charging Station sends a NotifyMonitoringReportRequest	16. The Test System responds with a NotifyMonitoringReportResponse .

Tool validations
<p>* Step 2: Message GetMonitoringReportResponse - status <i>Accepted</i></p> <p>* Step 6: Message SetVariableMonitoringResponse - setMonitoringResult[0].status <i>Accepted</i> - setMonitoringResult[0].type <i>Delta</i> - setMonitoringResult[0].component <preconfiguredMonitor.component> - setMonitoringResult[0].variable <preconfiguredMonitor.variable></p> <p>* Step 8: Message GetMonitoringReportResponse - status <i>Accepted</i></p> <p>* Step 9: Message NotifyMonitoringReportRequest Should contain monitor: - monitor.component <preconfiguredMonitor.component> - monitor.variable <preconfiguredMonitor.variable> - monitor.variableMonitoring.id <preconfiguredMonitor.variableMonitoring.id> - monitor.variableMonitoring.severity IF <preconfiguredMonitor.variableMonitoring.severity> < 9 THEN <preconfiguredMonitor.variableMonitoring.severity> + 1 ELSE 5 END IF</p> <p>* Step 15: Message NotifyMonitoringReportRequest Should contain monitor: - monitor.component <preconfiguredMonitor.component> - monitor.variable <preconfiguredMonitor.variable> - monitor.variableMonitoring.id <preconfiguredMonitor.variableMonitoring.id> - monitor.variableMonitoring.severity <preconfiguredMonitor.variableMonitoring.severity></p>

Tool validations
Post scenario validations: - All report parts have been received

TC_N_43_CS: Set Variable Monitoring - First SetMonitoringData and third SetMonitoringData are valid, but the second contains an out of range value

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 This test case assumes the Charging Station has a numeric Component Variable <Configured numeric delta component variable> and numeric Component Variable <Configured threshold monitor component variable>.

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetVariableMonitoringResponse	<p>1. The Test System sends a SetVariableMonitoringRequest with</p> <ul style="list-style-type: none"> - setMonitoringData[0].component = <Configured threshold monitor component variable> - setMonitoringData[0].variable = <Configured threshold monitor component variable> - setMonitoringData[0].value = <Configured threshold monitor component variable UpperThreshold trigger value> - setMonitoringData[0].type = UpperThreshold <ul style="list-style-type: none"> - setMonitoringData[1].component = <Configured numeric delta component variable> - setMonitoringData[1].variable = <Configured numeric delta component variable> - setMonitoringData[1].value = -1.0 - setMonitoringData[1].type = Delta <ul style="list-style-type: none"> - setMonitoringData[2].component = <Configured threshold monitor component variable> - setMonitoringData[2].variable = <Configured threshold monitor component variable> - setMonitoringData[2].value = <Configured threshold monitor component variable LowerThreshold trigger value> - setMonitoringData[2].type = LowerThreshold

Tool validations
<p>* Step 2:</p> <p>Message: SetVariableMonitoringResponse with (in arbitrary order):</p> <p>setMonitoringResult[1] = {</p> <ul style="list-style-type: none">- status = <i>Accepted</i>- type = <i>UpperThreshold</i>- statusInfo is absent or statusInfo.reasonCode = <i>"NoError"</i> <p>}</p> <p>setMonitoringResult[2] = {</p> <ul style="list-style-type: none">- status = <i>Rejected</i>- type = <i>Delta</i> <p>}</p> <p>setMonitoringResult[3] = {</p> <ul style="list-style-type: none">- status = <i>Accepted</i>- type = <i>LowerThreshold</i>- statusInfo is absent or statusInfo.reasonCode = <i>"NoError"</i> <p>}</p>
<p>Post scenario validations:</p> <ul style="list-style-type: none">- N/a

TC_N_44_CS: Clear Monitoring - Rejected

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
Reusable State: Test System Sends a GetMonitoringReportRequest, the CS then reports all existings 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 Test System sends a ClearVariableMonitoringRequest with <i>id monitor id from the Preparations</i>

Tool validations
* Step 2: Message ClearVariableMonitoringResponse - clearMonitoringResult[0].status Rejected
Post scenario validations: - N/a

TC_N_45_CS: Alert Event - Delta value exceeded

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** = <Configured numeric delta component variable>
- **setMonitoringData.variable** = <Configured numeric delta component variable Delta numeric trigger value>
- **setMonitoringData.value** = <Configured numeric delta component variable>
- **setMonitoringData.type** = Delta
- **setMonitoringData.severity** = 5

Set MonitoringLevel to 8

Notes: If componentVariable is set to "Power" or "Current", the value is set to 100.0

Reusable State:

N/a

Main (Test scenario)

Charging Station	CSMS
<u>Manual Action:</u> 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 <i>EnergyTransferStarted</i> or manually trigger the monitor.	
2. The Charging Station sends a NotifyEventRequest	3. The Test System responds with a NotifyEventResponse .
<u>Note(s):</u> - 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** <Configured numeric delta component variable>
- **eventData[0].variable** <Configured numeric delta component variable>
- **eventData[0].variableMonitoringId** <SetVariableMonitoringResponse.setMonitoringResult.id in preparation phase>

Post scenario validations:

- N/a

TC_N_47_CS: Get Monitoring report - Report all

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 <i>Delta</i> - Component "EVSE", Configured evse, variable "AvailabilityState", monitor type <i>Delta</i>
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a GetMonitoringReportResponse	1. The Test System sends a GetMonitoringReportRequest with monitoringCriteria omitted AND componentVariable omitted.
3. The Charging Station sends a NotifyMonitoringReportRequest	4. The Test System responds with a NotifyMonitoringReportResponse .
Note(s): - If tbc is <i>True</i> at Step 3 then step 3 and 4 will be repeated	

Tool validations
* Step 3: Message: NotifyMonitoringReportRequest - requestId = <Generated requestId> While tbc = <i>true</i> , Message: NotifyMonitoringReportRequest - monitor.variable = "AvailabilityState" - monitor.variableMonitoring.type = <i>Delta</i> - monitor.component_name = <i>ChargingStation</i> or <i>EVSE</i>
Post scenario validations: - All reports have been received

TC_N_48_CS: Alert Event - Variable monitoring on write only

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetVariablesResponse .	1. The Test System sends a SetVariablesRequest with component.name = <i>SecurityCtrlr</i> variable.name = <i>BasicAuthPassword</i> attributeValue = <i><Generated password with same length as the configured basicAuthPassword></i>
3. Execute Reusable State <i>Booted</i> . <u>Notes:</u> This step only needs to be executed when SetVariablesResponse status is <i>RebootRequired</i> .	
4. The Charging station sends a NotifyEventRequest	5. The Test System responds with a NotifyEventResponse .

Tool validations
* Step 2: Message SetVariablesResponse - status must be <i>Accepted</i> or <i>RebootRequired</i>
* Step 4: Message NotifyEventRequest - eventData[0].actualValue must be an empty string
Post scenario validations: - N/a

TC_N_61_CS: Alert Event - Variable monitoring on numeric

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 numeric 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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetVariablesResponse .	1. The Test System sends a SetVariablesRequest with component.name = OCPPCommCtrlr variable.name = OfflineThreshold attributeValue = Current Threshold + 1
3. Execute Reusable State <i>Booted</i> . <u>Notes:</u> This step only needs to be executed when SetVariablesResponse status is RebootRequired.	
<u>Notes:</u> The CS should not send a NotifyEvent as the delta monitor was not exceeded.	
5. The Charging Station responds with a SetVariablesResponse .	4. The Test System sends a SetVariablesRequest with component.name = OCPPCommCtrlr variable.name = OfflineThreshold attributeValue = Current Threshold + 2
6. Execute Reusable State <i>Booted</i> . <u>Notes:</u> This step only needs to be executed when SetVariablesResponse status is RebootRequired.	
7. The Charging station sends a NotifyEventRequest	8. The Test System responds with a NotifyEventResponse .

Tool validations
<p>* 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</p>
Post scenario validations: - N/a

TC_N_51_CS: Set Variable Monitoring - Modifying a VariableMonitor and trigger

Test case name	Set Variable Monitoring - Modifying a VariableMonitor and trigger
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 <Configured threshold monitor component variable> setMonitoringData.variable <Configured threshold monitor component variable> setMonitoringData.value <Configured threshold monitor component variable UpperThreshold non-trigger value> setMonitoringData.type UpperThreshold setMonitoringData.severity 5 Set MonitoringLevel to 8 <u>Notes:</u> If componentVariable is set to "Power" or "Current", the value is set to the configured maxLimit -1
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
<u>Notes:</u> 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 Test System sends a SetVariableMonitoringRequest with setMonitoringData.component <Configured threshold monitor component variable> setMonitoringData.variable <Configured threshold monitor component variable> setMonitoringData.id <SetVariableMonitoringResponse.setMonitoringResult.id in preparation phase> setMonitoringData.value <Configured threshold monitor component variable UpperThreshold trigger value> setMonitoringData.type UpperThreshold <u>Notes:</u> If componentVariable is set to "Power" or "Current", the value is set to 0.0
4. The Charging station sends a NotifyEventRequest	5. The Test System responds with a NotifyEventResponse .

Tool validations
<div>* Step 3: Message SetVariableMonitoringResponse<ul style="list-style-type: none">- setMonitoringResult[0].status <i>Accepted</i>- setMonitoringResult[0].type <i>UpperThreshold</i>- setMonitoringResult[0].severity <i>5</i>- setMonitoringResult[0].component <i><Configured threshold monitor component variable></i>- setMonitoringResult[0].variable <i><Configured threshold monitor component variable></i></div> <div>* Step 4: Message NotifyEventRequest<ul style="list-style-type: none">- eventData[0].trigger <i>Alerting</i>- eventData[0].actualValue <i>> <Configured threshold monitor component variable UpperThreshold non-trigger value></i></div>
<div>Post scenario validations:<ul style="list-style-type: none">- All report parts have been received</div>

TC_N_52_CS: Set Variable Monitoring - Removing a VariableMonitor

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. If <i>Power</i> or <i>Current</i> is used as the monitored variable, then power must flow when a transaction reaches EnergyTransfer, or else the monitor (power or current > 0) is not triggered.

Before (Preparations)

Configuration State:

N/a

Memory State:

Variable monitor is already set with:

setMonitoringData.component <Configured threshold monitor component variable>**setMonitoringData.variable** <Configured threshold monitor component variable>**setMonitoringData.value** <Configured threshold monitor component variable UpperThreshold trigger value>**setMonitoringData.type** UpperThreshold**setMonitoringData.severity** = 5

Set MonitoringLevel to 8

Notes: If componentVariable is set to "Power" or "Current", the value is set to 0.0

Reusable State:

N/a

Main (Test scenario)

Charging Station	CSMS
1. Execute Reusable State <i>EnergyTransferStarted</i> or manually trigger the monitor. <u>Notes:</u> 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.	
3. The Charging Station responds with a ClearVariableMonitoringResponse	2. The Test System sends a ClearVariableMonitoringRequest with id <monitoringId of monitor set in Memory State>
5. The Charging Station responds with a GetMonitoringReportResponse	4. The Test System sends a GetMonitoringReportRequest with componentVariable.component <Configured threshold monitor component variable> componentVariable.variable <Configured threshold monitor component variable> monitoringCriteria ThresholdMonitoring
6. Execute Reusable State <i>StopAuthorized</i> or manually trigger the monitor. <u>Notes:</u> 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.	
<u>Note:</u> The Charging Station should not send a request for the cleared monitor	

Tool validations
<div>* Step 1: Message NotifyEventRequest - eventData[0].trigger <i>Alerting</i> - eventData[0].component <i><Configured threshold monitor component variable></i> - eventData[0].variable <i><Configured threshold monitor component variable></i> - eventData[0].variableMonitoringId <i><monitoringId of monitor set in Memory State></i></div>
<div>* Step 3: Message ClearVariableMonitoringResponse - clearMonitoringResult[0].status <i>Accepted</i> AND - clearMonitoringResult[0].id <i><monitoringId of monitor set in Memory State></i></div>
<div>* Step 5: Message GetMonitoringReportResponse - getMonitoringResult[0].status <i>EmptyResultSet</i></div>
<div>Post scenario validations: - No NotifyEventRequest that variableMontioringId <i><monitoringId of monitor set in Memory State></i> is cleared, is sent.</div>

TC_N_53_CS: Alert Event - Persistent over reboot

Test case name	Alert Event - Persistent 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 <Configured threshold monitor component variable>**setMonitoringData.variable** <Configured threshold monitor component variable>**setMonitoringData.value** <Configured threshold monitor component variable UpperThreshold trigger value>**setMonitoringData.type** UpperThreshold

Reusable State:

Execute **Reusable State** *Booted*

Main (Test scenario)

Charging Station	CSMS
2. The Charging Station responds with a GetMonitoringReportResponse	1. The Test System sends a GetMonitoringReportRequest with monitoringCriteria <i>ThresholdMonitoring</i>
3. The Charging Station sends a NotifyMonitoringReportRequest	4. The Test System 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** <SetVariableMonitoringResponse.setMonitoringResult.id in preparation phase> -**monitor.variableMonitoring.type** UpperThreshold

Post scenario validations:

- All reports have been received

TC_N_56_CS: Alert Event - Delta value NOT numeric exceeded

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:

- **setMonitoringData[0].value** = 1
- **setMonitoringData[0].type** = *Delta*
- **setMonitoringData[0].severity** = 5
- **setMonitoringData[0].component** = <Configured non-numeric delta component variable>
- **setMonitoringData[0].variable** = <Configured non-numeric delta component variable>
- + Set MonitoringLevel to 8
- + Notes:
- Take a non-numeric component variable which can easily modified to trigger the alert.

Reusable State:

N/a

Main (Test scenario)

Charging Station	CSMS
<u>Manual Action:</u> Make sure the configured delta value has been exceeded	
1. The Charging Station sends a NotifyEventRequest	2. The Test System responds with a NotifyEventResponse .
<u>Note(s):</u> - 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** <Configured non-numeric delta component variable>
- **eventData[0].variable** <Configured non-numeric delta component variable>
- **eventData[0].variableMonitoringId** monitoringId of monitor set in Memory State

Post scenario validations:

- N/a

TC_N_100_CS: Retrieve Log Information - DataCollectorLog - Success

Test case name	Retrieve Log Information - DataCollectorLog - Success
Test case Id	TC_N_100_CS
Use case Id(s)	N01
Requirement(s)	N01.FR.01, N01.FR.02, N01.FR.07, N01.FR.08, N01.FR.09, N01.FR.13, N01.FR.24
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 of logType <i>DataCollectorLog</i> as described at the OCPP specification.
Prerequisite(s)	<ul style="list-style-type: none"> - Charging Station has log information available. - A diagnostics logging server has been setup supporting one of the file transfer protocols supported by the Charging Station (This is configured at the configuration variable FileTransferProtocols). - Charging station supports DataCollector (DataCollector.Available is <i>true</i>).

Before (Preparations)
Configuration State: <ul style="list-style-type: none"> - DataCollector.Enabled is <i>true</i> - DataCollector.DateTime[Start] is <i><current date/time>-2:00h</i> - DataCollector.DateTime[End] is <i><current date/time>+2:00h</i> - DataCollector.SampledMeasurands is <i>[Energy.Active.Import.Register]</i> - DataCollector.SamplingInterval is <i>1</i>
Memory State: N/a
Reusable State: EnergyTransferStarted

Main (Test scenario)	
Charging Station	CSMS
<u>Note(s)</u> : The Test System will wait for 10 seconds so the DataCollector is able to collect some data.	
2. The Charging Station responds with a GetLogResponse	1. The Test System sends a GetLogRequest with logType DataCollectorLog
<u>Note(s)</u> : - Charging Station is uploading log file	
3. The Charging Station sends a LogStatusNotificationRequest	4. The Test System responds with a LogStatusNotificationResponse
<u>Note(s)</u> : - Log file is uploaded	
5. The Charging Station sends a LogStatusNotificationRequest	6. The Test System responds with a LogStatusNotificationResponse

Tool validations
<p>* Step 2:</p> <p>Message GetLogResponse</p> <ul style="list-style-type: none">- status must be <i>Accepted</i>- filename must be <i><not omitted></i> <p>* Step 3:</p> <p>Message LogStatusNotificationRequest</p> <ul style="list-style-type: none">- status <i>Uploading</i>- requestId <i>Same Id as the GetLogRequest</i> <p>* Step 5:</p> <p>Message LogStatusNotificationRequest</p> <ul style="list-style-type: none">- status <i>Uploaded</i>- requestId <i>Same Id as the GetLogRequest</i>
<p>Post scenario validations:</p> <ul style="list-style-type: none">- File has been uploaded to remote location <i><Configured log location> + <GetLogResponse.filename of step 2></i>.

TC_N_101_CS: Retrieve Log Information - validations

Test case name	Retrieve Log Information - validations
Test case Id	TC_N_101_CS
Use case Id(s)	N01
Requirement(s)	N01.FR.30
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 will fail upload of a log of logType <i>DiagnosticsLog</i> to a redirected location.
Prerequisite(s)	<ul style="list-style-type: none"> - Charging Station has log information available. - OCPPCommCtrlr.FileTransferProtocols contains <i>HTTP</i> or <i>HTTPS</i> - A diagnostics logging server has been set up supporting for HTTP or HTTPS based on the file transfer protocols supported by the Charging Station (This is configured at the configuration variable OCPPCommCtrlr.FileTransferProtocols).

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a GetLogResponse	1. The Test System sends a GetLogRequest with logType <i>DiagnosticsLog</i> log.remoteLocation <Configured redirecting log location> + <configured log filename>
<u>Note(s):</u> - Charging Station is uploading log file	
3. The Charging Station sends a LogStatusNotificationRequest	4. The Test System responds with a LogStatusNotificationResponse

Tool validations
* Step 2: Message GetLogResponse - status must be <i>Accepted</i> - filename must be <configured log filename>
* Step 3: Message LogStatusNotificationRequest - status <i>UploadFailure</i> - requestId Same Id as the <i>GetLogRequest</i>
Post scenario validations: - No file has been uploaded to remote location <Configured redirecting log location> + <configured log filename>.

TC_N_102_CS: Retrieve Log Information - Authentication - HTTP

Test case name	Retrieve Log Information - Authentication - HTTP
Test case Id	TC_N_102_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, N01.FR.21 N01.FR.25
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 of logType <i>DiagnosticsLog</i> as described at the OCPP specification, using the HTTP protocol.
Prerequisite(s)	<ul style="list-style-type: none"> - Charging Station has log information available. - OCPPCommCtrlr.FileTransferProtocols contains <i>HTTP</i> - A diagnostics logging server has been set up for HTTP and accepts only specific userinfo credentials .

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a GetLogResponse	1. The Test System sends a GetLogRequest with logType <i>DiagnosticsLog</i> log.remoteLocation <Configured log location with userinfo> and set protocol to HTTP For example: http://username:password@my.server.net
<u>Note(s):</u> - Charging Station is uploading log file	
3. The Charging Station sends a LogStatusNotificationRequest	4. The Test System responds with a LogStatusNotificationResponse
<u>Note(s):</u> - Log file is uploaded	
5. The Charging Station sends a LogStatusNotificationRequest	6. The Test System responds with a LogStatusNotificationResponse

Tool validations
<p>* Step 2:</p> <p>Message GetLogResponse</p> <ul style="list-style-type: none">- status must be <i>Accepted</i>- filename must be <i><Not omitted AND not empty></i> <p>* Step 3:</p> <p>Message LogStatusNotificationRequest</p> <ul style="list-style-type: none">- status <i>Uploading</i>- requestId <i><Same Id as the GetLogRequest></i> <p>* Step 5:</p> <p>Message LogStatusNotificationRequest</p> <ul style="list-style-type: none">- status <i>Uploaded</i>- requestId <i><Same Id as the GetLogRequest></i>
<p>Post scenario validations:</p> <ul style="list-style-type: none">- File has been uploaded to remote location <i><Configured http log location> + <GetLogResponse.filename of step 2></i> using credentials mentioned in <i><Configured log location with userinfo></i>.

TC_N_103_CS: Retrieve Log Information - Authentication - HTTPS

Test case name	Retrieve Log Information - Authentication - HTTPS
Test case Id	TC_N_103_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, N01.FR.21 N01.FR.25
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 of logType <i>DiagnosticsLog</i> as described at the OCPP specification, using the HTTPS protocol.
Prerequisite(s)	<ul style="list-style-type: none"> - Charging Station has log information available. - Charging station supports HTTPS file transfer protocol (This is configured at the configuration variable FileTransferProtocols). - A diagnostics logging server has been set up for HTTPS and accepts only specific userinfo credentials .

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a GetLogResponse	1. The Test System sends a GetLogRequest with logType <i>DiagnosticsLog</i> log.remoteLocation <Configured log location with userinfo> and set protocol to HTTPS For example: https://username:password@my.server.net
Note(s): - Charging Station is uploading log file	
3. The Charging Station sends a LogStatusNotificationRequest	4. The Test System responds with a LogStatusNotificationResponse
Note(s): - Log file is uploaded	
5. The Charging Station sends a LogStatusNotificationRequest	6. The Test System responds with a LogStatusNotificationResponse

Tool validations
<p>* Step 2:</p> <p>Message GetLogResponse</p> <ul style="list-style-type: none">- status must be <i>Accepted</i>- filename must be <i><Not omitted AND not empty></i> <p>* Step 3:</p> <p>Message LogStatusNotificationRequest</p> <ul style="list-style-type: none">- status <i>Uploading</i>- requestId <i><Same Id as the GetLogRequest></i> <p>* Step 5:</p> <p>Message LogStatusNotificationRequest</p> <ul style="list-style-type: none">- status <i>Uploaded</i>- requestId <i><Same Id as the GetLogRequest></i>
<p>Post scenario validations:</p> <ul style="list-style-type: none">- File has been uploaded to remote location <i><Configured https log location> + <GetLogResponse.filename of step 2></i> using credentials mentioned in <i><Configured log location with userinfo></i>.

TC_N_104_CS: Get Monitoring report - TargetDeltaMonitoring

Test case name	Get Monitoring report - TargetDeltaMonitoring
Test case Id	TC_N_104_CS
Use case Id(s)	N02
Requirement(s)	N02.FR.01, N02.FR.03, N02.FR.04, N02.FR.06, N02.FR.09, N02.FR.10, N02.FR.22, N02.FR.23
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 for Component Variable <Configured numeric delta component variable>: - TargetDeltaRelative <Configured target delta relative value>
Memory State: N/a
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
2. Charging Station responds with: GetMonitoringReportResponse	1. Test System sends GetMonitoringReportRequest with: - requestId = <Generated requestId> - monitoringCriteria = { <i>ThresholdMonitoring</i> }
3. Charging Station responds with: NotifyMonitoringReportRequest	4. Test System sends NotifyMonitoringReportResponse
Step 3 and 4 occur when predefined or hardwired monitors exist on charging station, and are repeated as often as needed to report all configuration variables.	
6. Charging Station responds with: GetMonitoringReportResponse	5. Test System sends GetMonitoringReportRequest with: - requestId = <Generated requestId> - monitoringCriteria = { <i>DeltaMonitoring</i> }
7. Charging Station responds with: NotifyMonitoringReportRequest	8. Test System sends NotifyMonitoringReportResponse
Step 7 and 8 are repeated as often as needed to report all configuration variables.	

Tool validations
<p>* Step 2:</p> <p>Message: GetMonitoringReportResponse</p> <ul style="list-style-type: none"> - status = <i>EmptyResultSet</i> or <i>Accepted</i> <p>* Step 3: (If status = <i>Accepted</i>)</p> <p>Message: NotifyMonitoringReportRequest</p> <ul style="list-style-type: none"> - requestId = <i><Generated requestId></i> - generatedAt = <i><timestamp at charging station></i> - seqNo = 0 - monitor.variableMonitoring.type = <i>UpperThreshold</i> or <i>LowerThreshold</i> - monitor.variableMonitoring.eventNotificationType = <i><not omitted></i> <p>While tbc = <i>true</i></p> <p>Message: NotifyMonitoringReportRequest</p> <ul style="list-style-type: none"> - seqNo is incremented by 1 - monitor.variableMonitoring.type = <i>UpperThreshold</i> or <i>LowerThreshold</i> - monitor.variableMonitoring.eventNotificationType = <i><not omitted></i> <p>* Step 6:</p> <p>Message: GetMonitoringReportResponse</p> <ul style="list-style-type: none"> - status = <i>Accepted</i> - statusInfo is absent or statusInfo.reasonCode = <i>"NoError"</i> <p>* Step 7:</p> <p>Message: NotifyMonitoringReportRequest</p> <ul style="list-style-type: none"> - requestId = <i><Generated requestId></i> - generatedAt = <i><timestamp at charging station></i> - monitor.variableMonitoring.type = <i>TargetDeltaRelative</i> - monitor.variableMonitoring.eventNotificationType = <i><not omitted></i>
<p>Post scenario validations:</p> <p>N/A</p>

TC_N_105_CS: Set Variable Monitoring - Set Frequent Periodic Variable Monitoring - Periodic

Test case name	Set Variable Monitoring - Set Frequent Periodic Variable Monitoring - Periodic
Test case Id	TC_N_105_CS
Use case Id(s)	N11, N12, N13
Requirement(s)	N11.FR.01, N11.FR.02, N12.FR.01, N13.FR.01, N13.FR.03, N13.FR.04, N13.FR.06
System under test	Charging Station
Description	To give the CSMS the ability to request efficient frequent periodic monitoring of variables.
Purpose	To test that Charging Station supports configuring frequent periodic variable monitoring.
Prerequisite(s)	N/a.

Before (Preparations)
Configuration State: N/a
Memory State: This test requires the Monitoring Base to be set to <i>All</i> . - SetMonitoringBaseRequest with monitoringBase = <i>All</i> . This test requires the existence of a <i>Periodic</i> monitor on a (numerical) variable. It is not mandated which variables are required to be monitored. This test uses the variable "Power" of component "EVSE", which is most likely to be available for monitoring.
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
2. Charging Station responds with a SetVariableMonitoringResponse	1. Test System sends a SetVariableMonitoringRequest with setMonitoringData[0].value 1 setMonitoringData[0].type <i>Periodic</i> setMonitoringData[0].severity <Configured severity> setMonitoringData[0].component.name <Configured monitor component> setMonitoringData[0].component.evse.id <Configured evseld> setMonitoringData[0].variable.name <Configured monitor component variable> setMonitoringData[0].periodicEventStream.interval 10 setMonitoringData[0].periodicEventStream.values 30
3. Charging Station sends a OpenPeriodicEventStreamRequest	4. Test System responds with a OpenPeriodicEventStreamResponse with status <i>Accepted</i>
6. Charging Station response with a GetPeriodicEventStreamResponse	5. Test System sends a GetPeriodicEventStreamRequest
<u>Note:</u> Test System waits for 20 seconds	
7. Charging Station sends a NotifyPeriodicEventStream using RFC framework SEND	After 10 seconds
8 Charging Station sends a NotifyPeriodicEventStream using RFC framework SEND	After another 10 seconds

Main (Test scenario)	
10. Charging Station responds with a SetVariableMonitoringResponse	<p>9. Test System sends a SetVariableMonitoringRequest with setMonitoringData[0].id <i><SetVariableMonitoringResponse.setMonitoringResult[0].id of step 2></i></p> <p>setMonitoringData[0].value 1</p> <p>setMonitoringData[0].type <i>Periodic</i></p> <p>setMonitoringData[0].severity <i><Configured severity></i></p> <p>setMonitoringData[0].component.name <i><Configured monitor component></i></p> <p>setMonitoringData[0].component.evse.id <i><Configured evseld></i></p> <p>setMonitoringData[0].variable.name <i><Configured monitor component variable></i></p> <p>setMonitoringData[0].periodicEventStream <i><omitted></i></p>
11 Charging Station sends a NotifyPeriodicEventStream using RFC framework SEND	<i>This step may be omitted is buffer already empty</i>
12. Charging Station sends a ClosePeriodicEventStreamRequest	13. Test System responds with a ClosePeriodicEventStreamResponse
14. Charging Station sends a NotifyEventRequest	15. Test System responds with a NotifyEventResponse
<u>Note:</u> Step 14 and 15 will repeat every second	
<u>Note:</u> The Test System waits 20 seconds, before ending the testcase.	

Tool validations

* Step 2:

Message: **SetVariableMonitoringResponse**

- **setMonitoringResult[0].id** must be *<not omitted>*
- **setMonitoringResult[0].status** must be *Accepted*
- **setMonitoringResult[0].type** must be *Periodic*
- **setMonitoringResult[0].severity** must be *<Configured severity>*
- **setMonitoringResult[0].component.name** must be *<Configured monitor component>*
- **setMonitoringResult[0].component.evse.id** must be *<Configured evseId>*
- **setMonitoringResult[0].variable.name** must be *<Configured monitor component variable>*

* Step 3:

Message: **OpenPeriodicEventStreamRequest**

- **constantStreamData.id** must be *<not omitted>*
- **constantStreamData.variableMonitoringId** must be *<SetVariableMonitoringResponse.setMonitoringResult[0].id of step 2>*
- **constantStreamData.params.interval** must be *10*
- **constantStreamData.params.values** must be *30*

* Step 6:

Message: **GetPeriodicEventStreamResponse**

contains at least

- **constantStreamData[0].id** must be *<not omitted>*
- **constantStreamData[0].variableMonitoringId** must be *<SetVariableMonitoringResponse.setMonitoringResult[0].id of step 2>*
- **constantStreamData[0].params.interval** must be *10*
- **constantStreamData[0].params.values** must be *30*

* Step 7:

Message: **NotifyPeriodicEventStream**

- **id** must be *<OpenPeriodicEventStreamRequest.constantStreamData.id of step 3>*
- **data** array must contain 9 or 10 elements

* Step 8:

Message: **NotifyPeriodicEventStream**

- **id** must be *<OpenPeriodicEventStreamRequest.constantStreamData.id of step 3>*
- **data** array must contain 9 or 10 elements

* Step 10:

Message: **SetVariableMonitoringResponse**

- **setMonitoringResult[0].id** must be *<SetVariableMonitoringResponse.setMonitoringResult[0].id of step 2>*
- **setMonitoringResult[0].status** must be *Accepted*
- **setMonitoringResult[0].type** must be *Periodic*
- **setMonitoringResult[0].severity** must be *<Configured severity>*
- **setMonitoringResult[0].component.name** must be *<Configured monitor component>*
- **setMonitoringResult[0].component.evse.id** must be *<Configured evseId>*
- **setMonitoringResult[0].variable.name** must be *<Configured monitor component variable>*

* Step 11:

Message: **NotifyPeriodicEventStream**

- **id** must be *<OpenPeriodicEventStreamRequest.constantStreamData.id of step 3>*

* Step 12:

Message **ClosePeriodicEventStreamRequest**

- **id** *<OpenPeriodicEventStreamRequest.constantStreamData.id of step 3>*

*** Step 14:****Message: NotifyEventRequest**

- **generatedAt** must be *<not omitted>*
- **eventData[0].trigger** must be *Periodic*
- **eventData[0].actualValue** must be *<not omitted>*
- **eventData[0].variableMonitoringId** must be *<SetVariableMonitoringResponse.setMonitoringResult[0].id of step 2>*
- **eventData[0].severity** must be *<Configured severity>*
- **eventData[0].component.name** must be *<Configured monitor component>*
- **eventData[0].component.evse.id** must be *<Configured evseId>*
- **eventData[0].variable.name** must be *<Configured monitor component variable>*

Post scenario validations:

- * After step 12 no message of type **NotifyPeriodicEventStream** may be sent with
- **id** set to *<OpenPeriodicEventStreamRequest.constantStreamData.id of step 3>*

TC_N_106_CS: Set Variable Monitoring - Set Frequent Periodic Variable Monitoring - CSMS rejects stream

Test case name	Set Variable Monitoring - Set Frequent Periodic Variable Monitoring - CSMS rejects stream
Test case Id	TC_N_106_CS
Use case Id(s)	N11
Requirement(s)	N11.FR.01, N11.FR.02, N11.FR.07
System under test	Charging Station
Description	To give the CSMS the ability to request efficient frequent periodic monitoring of variables.
Purpose	To test that Charging Station falls back to NotifyEvent if a OpenPeriodicEventRequest is rejected by CSMS.
Prerequisite(s)	This test requires the existence of a <i>Periodic</i> monitor on a (numerical) variable. It is not mandated which variables are required to be monitored. This test uses the variable "Power" of component "EVSE", which is most likely to be available for monitoring.

Before (Preparations)
Configuration State: N/a
Memory State: This test requires the Monitoring Base to be set to <i>All</i> . - SetMonitoringBaseRequest with monitoringBase = <i>All</i> .
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
2. Charging Station responds with a SetVariableMonitoringResponse	1. Test System sends a SetVariableMonitoringRequest with setMonitoringData[0].value <Configured Clock Aligned MeterValues Interval> setMonitoringData[0].type <i>Periodic</i> setMonitoringData[0].severity <Configured severity> setMonitoringData[0].component.name <Configured threshold monitor component variable> setMonitoringData[0].component.evse.id <Configured evseld> setMonitoringData[0].variable.name <Configured threshold monitor component variable> setMonitoringData[0].periodicEventStream.interval 10 setMonitoringData[0].periodicEventStream.values 30
3. Charging Station sends a OpenPeriodicEventStreamRequest	4. Test System responds with a OpenPeriodicEventStreamResponse with status <i>Rejected</i>
5. Charging Station sends a NotifyEventRequest	6. Test System responds with a NotifyEventResponse
<u>Note:</u> Step 5 and 6 will repeat every second	
<u>Note:</u> The Test System waits 11 seconds, before ending the testcase.	

Tool validations

* Step 2:

Message: **SetVariableMonitoringResponse**

- **setMonitoringResult[0].id** must be *<not omitted>*
- **setMonitoringResult[0].status** must be *Accepted*
- **setMonitoringResult[0].type** must be *Periodic*
- **setMonitoringResult[0].severity** must be *<Configured severity>*
- **setMonitoringResult[0].component.name** must be *<Configured threshold monitor component variable>*
- **setMonitoringResult[0].component.evse.id** must be *<Configured evseld>*
- **setMonitoringResult[0].variable.name** must be *<Configured threshold monitor component variable>*

No message of type **NotifyPeriodicEventStream** shall be sent with

- **id** set to *<SetVariableMonitoringResponse.setMonitoringResult[0].id>*

* Step 3:

Message: **OpenPeriodicEventStreamRequest**

- **constantStreamData.id** must be *<not omitted>*
- **constantStreamData.variableMonitoringId** must be *<SetVariableMonitoringResponse.setMonitoringResult[0].id of step 2>*
- **constantStreamData.params.interval** must be *10*
- **constantStreamData.params.values** must be *30*

* Step 5:

Message: **NotifyEventRequest**

- **generatedAt** must be *<not omitted>*
- **eventData[0].trigger** must be *Periodic*
- **eventData[0].actualValue** must be *<not omitted>*
- **eventData[0].variableMonitoringId** must be *<SetVariableMonitoringResponse.setMonitoringResult[0].id of step 2>*
- **eventData[0].severity** must be *<Configured severity>*
- **eventData[0].component.name** must be *<Configured threshold monitor component variable>*
- **eventData[0].component.evse.id** must be *<Configured evseld>*
- **eventData[0].variable.name** must be *<Configured threshold monitor component variable>*

Post scenario validations:

No message of type **NotifyPeriodicEventStream** shall be sent with

- **id** set to *<SetVariableMonitoringResponse.setMonitoringResult[0].id of step 2>*

TC_N_108_CS: Set Variable Monitoring - Close Periodic Event Streams

Test case name	Set Variable Monitoring - Close Periodic Event Streams
Test case Id	TC_N_108_CS
Use case Id(s)	N11, N13
Requirement(s)	N11.FR.01, N11.FR.02, N13.FR.01, N13.FR.03, N13.FR.05
System under test	Charging Station
Description	To give the CSMS the ability to request efficient frequent periodic monitoring of variables.
Purpose	To test that Charging Station closes the periodic event stream when the monitor is cleared.
Prerequisite(s)	N/a.

Before (Preparations)

Configuration State:

N/a

Memory State:

This test requires the Monitoring Base to be set to *All*.

- **SetMonitoringBaseRequest** with **monitoringBase** = *All*.

This test requires the existence of a *Periodic* monitor on a (numerical) variable. It is not mandated which variables are required to be monitored. This test uses the variable "Power" of component "EVSE", which is most likely to be available for monitoring.

Reusable State(s): N/a

Main (Test scenario)

Charging Station	CSMS
2. Charging Station responds with a SetVariableMonitoringResponse	1. Test System sends a SetVariableMonitoringRequest with setMonitoringData[0].value 1 setMonitoringData[0].type <i>Periodic</i> setMonitoringData[0].severity <Configured severity> setMonitoringData[0].component.name <Configured monitor component> setMonitoringData[0].component.evse.id <Configured evseld> setMonitoringData[0].variable.name <Configured monitor component variable> setMonitoringData[0].periodicEventStream.interval 10 setMonitoringData[0].periodicEventStream.values 30
3. Charging Station sends a OpenPeriodicEventStreamRequest	4. Test System responds with a OpenPeriodicEventStreamResponse with status <i>Accepted</i>
6. Charging Station response with a ClearVariableMonitoringResponse	5. Test System sends a ClearVariableMonitoringRequest with id <SetVariableMonitoringResponse.setMonitoringResult[0].id of step 2>
7. Charging Station sends a ClosePeriodicEventStreamRequest	8. Test System responds with a ClosePeriodicEventStreamResponse

Note: The Test System waits 20 seconds, before ending the testcase.

Tool validations
<p>* Step 2:</p> <p>Message: SetVariableMonitoringResponse</p> <ul style="list-style-type: none"> - setMonitoringResult[0].id must be <i><not omitted></i> - setMonitoringResult[0].status must be <i>Accepted</i> - setMonitoringResult[0].type must be <i>Periodic</i> - setMonitoringResult[0].severity must be <i><Configured severity></i> - setMonitoringResult[0].component.name must be <i><Configured monitor component></i> - setMonitoringResult[0].component.evse.id must be <i><Configured evseld></i> - setMonitoringResult[0].variable.name must be <i><Configured monitor component variable></i> <p>* Step 3:</p> <p>Message: OpenPeriodicEventStreamRequest</p> <ul style="list-style-type: none"> - constantStreamData.id must be <i><not omitted></i> - constantStreamData.variableMonitoringId must be <i><SetVariableMonitoringResponse.setMonitoringResult[0].id of step 2></i> - constantStreamData.params.interval must be <i>10</i> - constantStreamData.params.values must be <i>30</i> <p>* Step 6:</p> <p>Message: ClearVariableMonitoringResponse</p> <ul style="list-style-type: none"> - clearMonitoringResult[0].status must be <i>Accepted</i> - clearMonitoringResult[0].id must be <i><SetVariableMonitoringResponse.setMonitoringResult[0].id of step 2></i> <p>* Step 7:</p> <p>Message ClosePeriodicEventStreamRequest</p> <ul style="list-style-type: none"> - id <i><OpenPeriodicEventStreamRequest.constantStreamData.id of step 3></i>
<p>Post scenario validations:</p> <p>* After step 9 no message of type NotifyPeriodicEventStream may be sent with</p> <ul style="list-style-type: none"> - id set to <i><OpenPeriodicEventStreamRequest.constantStreamData.id of step 3></i>

TC_N_109_CS: Set Variable Monitoring - Adjust Periodic Event Streams

Test case name	Set Variable Monitoring - Adjust Periodic Event Streams
Test case Id	TC_N_109_CS
Use case Id(s)	N11, N14, N15
Requirement(s)	N11.FR.01, N11.FR.02, N14.FR.03, N15.FR.01, N15.FR.04
System under test	Charging Station
Description	To adjust the transmission rate of a periodic event stream
Purpose	To test that Charging Station supports adjust periodic event streams.
Prerequisite(s)	N/a.

Before (Preparations)

Configuration State:

N/a

Memory State:

This test requires the Monitoring Base to be set to *All*.

- **SetMonitoringBaseRequest** with **monitoringBase** = *All*.

This test requires the existence of a *Periodic* monitor on a (numerical) variable. It is not mandated which variables are required to be monitored. This test uses the variable "Power" of component "EVSE", which is most likely to be available for monitoring.

Reusable State(s): N/a

Main (Test scenario)

Charging Station	CSMS
2. Charging Station responds with a SetVariableMonitoringResponse	1. Test System sends a SetVariableMonitoringRequest with setMonitoringData[0].value 1 setMonitoringData[0].type <i>Periodic</i> setMonitoringData[0].severity <Configured severity> setMonitoringData[0].component.name <Configured monitor component> setMonitoringData[0].component.evse.id <Configured evseld> setMonitoringData[0].variable.name <Configured monitor component variable> setMonitoringData[0].periodicEventStream.interval 10
3. Charging Station sends a OpenPeriodicEventStreamRequest	4. Test System responds with a OpenPeriodicEventStreamResponse with status <i>Accepted</i>
5. Charging Station sends a NotifyPeriodicEventStream using RFC framework SEND	After 10 seconds
6. Charging Station sends a NotifyPeriodicEventStream using RFC framework SEND	After 10 seconds
7. Charging Station sends a NotifyPeriodicEventStream using RFC framework SEND	After 10 seconds
9. Charging Station responds with a AdjustPeriodicEventStreamResponse	8. Test System sends a AdjustPeriodicEventStreamRequest with id <OpenPeriodicEventStreamRequest.constantStreamData.id of step 3> params.interval 15

Main (Test scenario)	
10. Charging Station sends a NotifyPeriodicEventStream using RFC framework SEND	After 15 seconds
11. Charging Station sends a NotifyPeriodicEventStream using RFC framework SEND	After 15 seconds
12. Charging Station sends a NotifyPeriodicEventStream using RFC framework SEND	After 15 seconds

Tool validations
<p>* Step 2:</p> <p>Message: SetVariableMonitoringResponse</p> <ul style="list-style-type: none"> - setMonitoringResult[0].id must be <i><not omitted></i> - setMonitoringResult[0].status must be <i>Accepted</i> - setMonitoringResult[0].type must be <i>Periodic</i> - setMonitoringResult[0].severity must be <i><Configured severity></i> - setMonitoringResult[0].component.name must be <i><Configured monitor component></i> - setMonitoringResult[0].component.evse.id must be <i><Configured evseld></i> - setMonitoringResult[0].variable.name must be <i><Configured monitor component variable></i> <p>* Step 3:</p> <p>Message: OpenPeriodicEventStreamRequest</p> <ul style="list-style-type: none"> - constantStreamData.id must be <i><not omitted></i> - constantStreamData.variableMonitoringId must be <i><SetVariableMonitoringResponse.setMonitoringResult[0].id of step 2></i> - constantStreamData.params.interval must be <i>10</i> <p>* Step 5:</p> <p>Message: NotifyPeriodicEventStream</p> <ul style="list-style-type: none"> - id must be <i><OpenPeriodicEventStreamRequest.constantStreamData.id of step 3></i> <p>* Step 6:</p> <p>Message: NotifyPeriodicEventStream shall be sent about 10 seconds after step 5</p> <ul style="list-style-type: none"> - id must be <i><OpenPeriodicEventStreamRequest.constantStreamData.id of step 3></i> <p>* Step 7:</p> <p>Message: NotifyPeriodicEventStream shall be sent about 10 seconds after step 6</p> <ul style="list-style-type: none"> - id must be <i><OpenPeriodicEventStreamRequest.constantStreamData.id of step 3></i> <p>* Step 9:</p> <p>Message: AdjustPeriodicEventStreamResponse</p> <ul style="list-style-type: none"> - status must be <i>Accepted</i> <p>* Step 10:</p> <p>Message: NotifyPeriodicEventStream</p> <ul style="list-style-type: none"> - id must be <i><OpenPeriodicEventStreamRequest.constantStreamData.id of step 3></i> <p>* Step 11:</p> <p>Message: NotifyPeriodicEventStream shall be sent about 15 seconds after step 10</p> <ul style="list-style-type: none"> - id must be <i><OpenPeriodicEventStreamRequest.constantStreamData.id of step 3></i> <p>* Step 12:</p> <p>Message: NotifyPeriodicEventStream shall be sent about 15 seconds after step 11</p> <ul style="list-style-type: none"> - id must be <i><OpenPeriodicEventStreamRequest.constantStreamData.id of step 3></i> <p>Post scenario validations: N/a</p>

O Display Message

TC_O_01_CS: Set Display Message - Success

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetDisplayMessageResponse	1. The Test System sends a SetDisplayMessageRequest with message.id <Generated displayMessageId> message.priority <Configured priority> message.display <Configured display component variable> (Omitted when left empty)
Note(s): - The display message is displayed as configured	
4. The Charging Station responds with a GetDisplayMessagesResponse	3. The Test System sends a GetDisplayMessagesRequest with id <Generated displayMessageId> requestId <Generated requestId>
5. The Charging Station sends a NotifyDisplayMessagesRequest	6. The Test System responds with a NotifyDisplayMessagesResponse .

Tool validations
<p>* Step 2: Message SetDisplayMessageResponse - status Accepted</p> <p>* Step 4: Message GetDisplayMessagesResponse - status Accepted</p> <p>* Step 5: Message NotifyDisplayMessagesRequest - requestId <RequestId sent in step 3> - id <Generated id> - priority <Configured Priority> - message.format <Configured format> - message.content <Configured content></p>

Tool validations
Post scenario validations: - N/a

TC_O_02_CS: Get all Display Messages - Success

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: SetDisplayMessage
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a GetDisplayMessagesResponse	1. The Test System sends a GetDisplayMessagesRequest requestId <Generated requestId>
3. The Charging Station sends a NotifyDisplayMessagesRequest	4. The Test System responds with a NotifyDisplayMessagesResponse .
<u>Note(s):</u> - If tbc is True at Step 3 then step 3 and 4 will be repeated	

Tool validations
<p>* Step 2: Message GetDisplayMessagesResponse - status Accepted</p> <p>* Step 3: Message NotifyDisplayMessagesRequest - requestId <Generated requestId></p>
<p>Post scenario validations: - All messages have been received</p>

TC_O_03_CS: Get all Display Messages - No DisplayMessages configured

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a GetDisplayMessagesResponse	1. The Test System sends a GetDisplayMessagesRequest

Tool validations
* Step 2: Message GetDisplayMessagesResponse - status <i>Unknown</i>
Post scenario validations: - N/a

TC_O_04_CS: Clear Display Message - Success

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: SetDisplayMessage
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a ClearDisplayMessageResponse	1. The Test System sends a ClearDisplayMessageRequest with id <Generated displayMessageId>
4. The Charging Station responds with a GetDisplayMessagesResponse	3. The Test System sends a GetDisplayMessagesRequest with id <Generated displayMessageId>

Tool validations
<p>* Step 2: Message ClearDisplayMessageResponse - status <i>Accepted</i></p> <p>* Step 4: Message: GetDisplayMessagesResponse - status must be <i>Unknown</i></p>
Post scenario validations: - N/a

TC_O_05_CS: Clear Display Message - Unknown Key

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a ClearDisplayMessageResponse	1. The Test System sends a ClearDisplayMessageRequest with id <Generated displayMessageId>

Tool validations
* Step 2: Message ClearDisplayMessageResponse - status <i>Unknown</i>
Post scenario validations: - N/a

TC_O_06_CS: Set Display Message - Specific transaction - Success

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
Reusable State: State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetDisplayMessageResponse	1. The Test System sends a SetDisplayMessageRequest with message.id <Generated displayMessageId> AND message.transactionId <Configured transactionId> AND message.priority <Configured Priority AND message.display <Configured display component variable> (Omitted when left empty)
Note(s): - The display message is displayed as configured	
3. Execute Reusable State <i>StopAuthorized</i>	
4. Execute Reusable State <i>EVConnectedPostSession</i>	
5. Execute Reusable State <i>EVDisconnected</i>	
6. Execute Reusable State <i>ParkingBayUnoccupied</i>	
Note(s): - The display message is not displayed anymore	
8. The Charging Station responds with a GetDisplayMessagesResponse	7. The Test System sends a GetDisplayMessagesRequest with id <Generated displayMessageId>

Tool validations
* Step 1: Message: SetDisplayMessageResponse - status must be <i>Accepted</i>
* Step 8: Message: GetDisplayMessagesResponse - status must be <i>Unknown</i>
Post scenario validations: N/a

TC_O_07_CS: Get a Specific Display Message - Id

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 the current list of messages, so the CSO knows which messages are (still) configured.
Purpose	To verify if the Charging Station is able to respond to 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: SetDisplayMessage
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a GetDisplayMessagesResponse	1. The Test System sends a GetDisplayMessagesRequest with id <Generated displayMessageld> requestId <Generated requestId>
3. The Charging Station sends a NotifyDisplayMessagesRequest	4. The Test System responds with a NotifyDisplayMessagesResponse .
<u>Note(s):</u> - 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

TC_O_08_CS: Get a Specific Display Message - Priority

Test case name	Get a Specific Display Message - Priority
Test case Id	TC_O_08_CS
Use case Id(s)	004
Requirement(s)	004_FR_01, 004_FR_03, 004_FR_04, 004_FR_05, 004_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: SetDisplayMessage
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a GetDisplayMessagesResponse	1. The Test System sends a GetDisplayMessagesRequest with priority <Configured display_message_priority> requestId <Generated requestId>
3. The Charging Station sends a NotifyDisplayMessagesRequest	4. The Test System responds with a NotifyDisplayMessagesResponse .
<u>Note(s):</u> - 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

TC_O_09_CS: Get a Specific Display Message - State

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 the current list of messages, so the CSO knows which messages are (still) configured.
Purpose	To verify if the Charging Station is able to respond to 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: SetDisplayMessage with state <Configured Message state>
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a GetDisplayMessagesResponse	1. The Test System sends a GetDisplayMessagesRequest with state <Configured display_message_state> requestId <Generated requestId>
3. The Charging Station sends a NotifyDisplayMessagesRequest	4. The Test System responds with a NotifyDisplayMessagesResponse .
<u>Note(s):</u> - 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

TC_O_10_CS: Set Display Message - Specific transaction - UnknownTransaction

Test case name	Set Display Message - Specific transaction - UnknownTransaction
Test case Id	TC_O_10_CS
Use case Id(s)	002
Requirement(s)	002_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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetDisplayMessageResponse	1. The Test System sends a SetDisplayMessageRequest with message.id <Generated displayMessageId> AND message.transactionId <Generated transactionId> AND message.priority <Configured Priority AND message.display <Configured display component variable> (Omitted when left empty)

Tool validations
* Step 2: Message SetDisplayMessageResponse - status <i>UnknownTransaction</i>
Post scenario validations: - N/a

TC_O_11_CS: Get a Specific Display Message - Unknown parameters

Test case name	Get a Specific Display Message - Unknown parameters
Test case Id	TC_O_11_CS
Use case Id(s)	004
Requirement(s)	004_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: SetDisplayMessage
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a GetDisplayMessagesResponse	1. The Test System sends a GetDisplayMessagesRequest with id <Other generated messageId>

Tool validations
* Step 2: Message GetDisplayMessagesResponse - status <i>Unknown</i>
Post scenario validations: - N/a

TC_O_12_CS: Set Display Message - Replace DisplayMessage

Test case name	Set Display Message - Replace DisplayMessage
Test case Id	TC_O_12_CS
Use case Id(s)	006
Requirement(s)	006_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: SetDisplayMessage
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetDisplayMessageResponse	1. The Test System sends a SetDisplayMessageRequest with message.id <Generated displayMessageId from before (preperation) steps> message.message.content <Different message to indicate the message was replaced>
<u>Note(s):</u> - The display message is replaced by a new one.	

Tool validations
* Step 2: Message SetDisplayMessageResponse - status <i>Accepted</i>
Post scenario validations: - N/a

TC_O_13_CS: Set Display Message - Display message at StartTime

Test case name	Set Display Message - Display message at StartTime
Test case Id	TC_O_13_CS
Use case Id(s)	001
Requirement(s)	001_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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetDisplayMessageResponse	1. The Test System sends a SetDisplayMessageRequest with message.id <Generated displayMessageId> message.priority <Configured Priority> message.startDateTime <Current dateTime + Configured Start Date Time Offset> message.display <Configured display component variable> (Omitted when left empty)
4. The Charging Station responds with a GetDisplayMessagesResponse	3. The Test System sends a GetDisplayMessagesRequest with id <Generated displayMessageId>
5. The Charging Station sends a NotifyDisplayMessagesRequest	6. The Test System responds with a NotifyDisplayMessagesResponse .

Note(s):

- If **tbc** is True at Step 5 then step 5 and 6 will be repeated
- Wait till <Configured Start Date Time Offset> seconds have passed
- The display message should be displayed after <Configured Start Date Time Offset> seconds.

Tool validations
<p>* Step 2: Message SetDisplayMessageResponse - status Accepted</p> <p>* Step 4: Message GetDisplayMessagesResponse - status Accepted</p> <p>* Step 5: Message NotifyDisplayMessagesRequest - requestId <Generated requestId> - startDateTime <Should not be Omitted.></p>

Tool validations
Post scenario validations: - N/a

TC_O_14_CS: Set Display Message - Remove message after EndTime

Test case name	Set Display Message - Remove message after EndTime
Test case Id	TC_O_14_CS
Use case Id(s)	001
Requirement(s)	001_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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetDisplayMessageResponse	1. The Test System sends a SetDisplayMessageRequest with message.id <Generated displayMessageId> message.priority <Configured Priority> message.endDateTime <Current dateTime + Configured End Date Time Offset> message.display <Configured display component variable> (Omitted when left empty)
4. The Charging Station responds with a GetDisplayMessagesResponse	3. The Test System sends a GetDisplayMessagesRequest with id <Generated displayMessageId>
5. The Charging Station sends a NotifyDisplayMessagesRequest	6. The Test System 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 <Configured End Date Time Offset> seconds have passed - The display message is displayed and removed after <Configured End Date Time Offset> seconds.	
8. The Charging Station responds with a GetDisplayMessagesResponse	7. The Test System sends a GetDisplayMessagesRequest with id <Generated displayMessageId> requestId <Generated requestId>

Tool validations
<div>* Step 2: Message SetDisplayMessageResponse - status <i>Accepted</i></div> <div>* Step 4: Message GetDisplayMessagesResponse - status <i>Accepted</i></div> <div>* Step 5: Message NotifyDisplayMessagesRequest - requestId <i><Generated requestId></i> - endDateTime <i><Should not be Omitted.></i></div> <div>* Step 8: Message GetDisplayMessagesResponse - status <i>Unknown</i></div>
<div>Post scenario validations: - N/a</div>

TC_O_17_CS: Set Display Message - NotSupportedPriority

Test case name	Set Display Message - NotSupportedPriority
Test case Id	TC_O_17_CS
Use case Id(s)	001
Requirement(s)	001.FR.01, 002.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

Reusable State:

N/a

Main (Test scenario)

Charging Station	CSMS
2. The Charging Station responds with a SetDisplayMessageResponse	<p>1. The Test System sends a SetDisplayMessageRequest with</p> <p>message.id <Generated displayMessageId></p> <p>message.priority <Configured unsupported_display_message_priority></p> <p>message.display <Configured display component variable></p> <p>(Omitted when left empty)</p>

Tool validations

* Step 2:

Message SetDisplayMessageResponse

- **status** NotSupportedPriority

Post scenario validations:

- N/a

TC_O_18_CS: Set Display Message - NotSupportedState

Test case name	Set Display Message - NotSupportedState
Test case Id	TC_O_18_CS
Use case Id(s)	001
Requirement(s)	001_FR_02, 002.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 correctly 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

Reusable State:

N/a

Main (Test scenario)

Charging Station	CSMS
2. The Charging Station responds with a SetDisplayMessageResponse	<p>1. The Test System sends a SetDisplayMessageRequest with</p> <p>message.id <Generated displayMessageId></p> <p>message.state <Configured unsupported_display_message_state></p> <p>message.display <Configured display component variable></p> <p>(Omitted when left empty)</p>

Tool validations

* Step 2:

Message **SetDisplayMessageResponse**- **status** NotSupportedState

Post scenario validations:

- N/a

TC_O_19_CS: Set Display Message - NotSupportedMessageFormat

Test case name	Set Display Message - NotSupportedMessageFormat
Test case Id	TC_O_19_CS
Use case Id(s)	001
Requirement(s)	001_FR_03, 002.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 correctly 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

Reusable State:

N/a

Main (Test scenario)

Charging Station	CSMS
2. The Charging Station responds with a SetDisplayMessageResponse	<p>1. The Test System sends a SetDisplayMessageRequest with</p> <p>message.id <Generated displayMessageId></p> <p>message.message.format <Configured Unsupported Message Format></p> <p>message.display <Configured display component variable></p> <p>(Omitted when left empty)</p>

Tool validations

* Step 2:

Message SetDisplayMessageResponse

- **status** NotSupportedMessageFormat

Post scenario validations:

- N/a

TC_O_20_CS: Set Display Message - Persistent over reboot

Test case name	Set Display Message - Persistent over reboot
Test case Id	TC_O_20_CS
Use case Id(s)	001
Requirement(s)	001_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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetDisplayMessageResponse	1. The Test System sends a SetDisplayMessageRequest with message.id <Generated displayMessageId> message.priority <Configured Priority> message.display <Configured display component variable> (Omitted when left empty)
3. Execute Reusable State <i>Booted</i>	
5. The Charging Station responds with a GetDisplayMessagesResponse	4. The Test System sends a GetDisplayMessagesRequest with id <Generated displayMessageId> requestId <Generated requestId>
6. The Charging Station sends a NotifyDisplayMessagesRequest	7. The Test System responds with a NotifyDisplayMessagesResponse .
Note(s): - If tlc is True at Step 5 then step 5 and 6 will be repeated	

Tool validations
<p>* Step 2: Message SetDisplayMessageResponse - status Accepted</p> <p>* Step 5: Message GetDisplayMessagesResponse - status Accepted</p> <p>* Step 6: Message NotifyDisplayMessagesRequest - requestId <RequestId sent in step 4> - id <Generated id> - priority <Configured Priority> - message.format <Configured format> - message.content <Configured content></p>

Tool validations
Post scenario validations: - N/a

TC_O_22_CS: Set Display Message - Multiple In front priority

Test case name	Set Display Message - Multiple In front priority
Test case Id	TC_O_22_CS
Use case Id(s)	001
Requirement(s)	001_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

Reusable State:

N/a

Main (Test scenario)

Charging Station	CSMS
2. The Charging Station responds with a SetDisplayMessageResponse	1. The Test System sends a SetDisplayMessageRequest with message.id <Generated displayMessageId> message.priority InFront message.display <Configured display component variable> (Omitted when left empty)
4. The Charging Station responds with a SetDisplayMessageResponse	3. The Test System sends a SetDisplayMessageRequest with message.id <Generated displayMessage2Id> message.priority InFront message.display <Configured display component variable> (Omitted when left empty)
6. The Charging Station responds with a GetDisplayMessagesResponse	5. The Test System sends a GetDisplayMessagesRequest with id <Generated displayMessageId> requestId <Generated requestId>
7. The Charging Station sends a NotifyDisplayMessagesRequest	8. The Test System responds with a NotifyDisplayMessagesResponse .
10. The Charging Station responds with a GetDisplayMessagesResponse	9. The Test System sends a GetDisplayMessagesRequest with id <Generated displayMessage2Id> requestId <Generated requestId>
11. The Charging Station sends a NotifyDisplayMessagesRequest	12. The Test System 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	

Tool validations
<div>* Step 2: Message SetDisplayMessageResponse - status <i>Accepted</i></div> <div>* Step 4: Message SetDisplayMessageResponse - status <i>Accepted</i></div> <div>* Step 6: Message GetDisplayMessagesResponse - status <i>Accepted</i></div> <div>* Step 7: Message NotifyDisplayMessagesRequest - requestId <i><Generated requestId></i></div> <div>* Step 10: Message GetDisplayMessagesResponse - status <i>Accepted</i></div> <div>* Step 11: Message NotifyDisplayMessagesRequest - requestId <i><Generated requestId></i></div>
<div>Post scenario validations: - N/a</div>

TC_O_24_CS: Set Display Message - Second Alwaysfront priority

Test case name	Set Display Message - Second Alwaysfront priority
Test case Id	TC_O_24_CS
Use case Id(s)	001
Requirement(s)	001_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

Reusable State:
N/a

Main (Test scenario)

Charging Station	CSMS
2. The Charging Station responds with a SetDisplayMessageResponse	1. The Test System sends a SetDisplayMessageRequest with message.id <Generated displayMessageId> message.priority AlwaysFront
4. The Charging Station responds with a SetDisplayMessageResponse	3. The Test System sends a SetDisplayMessageRequest with message.id <Configured displayMessage2Id> message.priority AlwaysFront
6. The Charging Station responds with a GetDisplayMessagesResponse	5. The Test System sends a GetDisplayMessagesRequest with id <Configured displayMessage2Id>
7. The Charging Station sends a NotifyDisplayMessagesRequest	8. The Test System responds with a NotifyDisplayMessagesResponse .

Note(s):

- If **tbc** is True at Step 7 then step 7 and 8 will be repeated
- The message from step 1 is NOT displayed anymore and is replaced by the message from step 5.

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>

Tool validations
Post scenario validations: - N/a

TC_O_27_CS: Set Display Message - Specific transaction - Display message at StartTime

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
Reusable State: State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetDisplayMessageResponse	1. The Test System sends a SetDisplayMessageRequest with message.id <Generated displayMessageId> message.priority <Configured Priority> message.startDateTime <Current dateTime + Configured Start Date Time Offset> message.display <Configured display component variable> (Omitted when left empty)
<u>Note(s):</u> - The display message is not yet displayed. - Waiting <Configured Start Date Time Offset> seconds. - The display message is displayed after <Configured Start Date Time Offset> seconds.	
3. Execute Reusable State <i>StopAuthorized</i>	
4. Execute Reusable State <i>EVConnectedPostSession</i>	
5. Execute Reusable State <i>EVDisconnected</i>	
6. Execute Reusable State <i>ParkingBayUnoccupied</i>	
<u>Note(s):</u> - The display message is not displayed anymore	
8. The Charging Station responds with a GetDisplayMessagesResponse	7. The Test System sends a GetDisplayMessagesRequest with id <Generated displayMessageId>

Tool validations
* Step 2: Message SetDisplayMessageResponse - status <i>Accepted</i> * Step 8: Message: GetDisplayMessagesResponse - status <i>Unknown</i>

Tool validations
Post scenario validations: - N/a

TC_O_28_CS: Set Display Message - Specific transaction - Remove message after EndTime

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
Reusable State: State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetDisplayMessageResponse	1. The Test System sends a SetDisplayMessageRequest with message.id <Generated displayMessageId> message.transactionId <Generated transactionId> message.priority <Configured Priority> message.endDateTime <Current dateTime + Configured End Date Time Offset> message.display <Configured display component variable> (Omitted when left empty)
<u>Note(s):</u> - The display message should be displayed. - Waiting <Configured End Date Time Offset> seconds. - The display message is not being displayed anymore after <Configured End Date Time Offset> seconds.	
4. The Charging Station responds with a GetDisplayMessagesResponse	3. The Test System sends a GetDisplayMessagesRequest with id <Generated displayMessageId>

Tool validations
* Step 2: Message SetDisplayMessageResponse - status <i>Accepted</i> * Step 4: Message GetDisplayMessagesResponse - status <i>Unknown</i>
Post scenario validations: - N/a

TC_O_30_CS: Set Display Message - Specific transaction - Multiple In front priority

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
Reusable State: State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetDisplayMessageResponse	1. The Test System sends a SetDisplayMessageRequest with message.id <Generated displayMessageId> AND message.transactionId <Received transactionId> AND message.priority <i>InFront</i> AND message.display <Configured display component variable> (Omitted when left empty)
4. The Charging Station responds with a SetDisplayMessageResponse	3. The Test System sends a SetDisplayMessageRequest with message.id <Generated displayMessageId2> AND message.transactionId <Received transactionId> AND message.priority <i>InFront</i> AND message.display <Configured display component variable> (Omitted when left empty)
Note(s): - The display messages are displayed as configured	
6. The Charging Station responds with a GetDisplayMessagesResponse	5. The Test System sends a GetDisplayMessagesRequest with id <Generated displayMessageId> requestId <Generated requestId>
7. The Charging Station sends a NotifyDisplayMessagesRequest	8. The Test System responds with a NotifyDisplayMessagesResponse .
10. The Charging Station responds with a GetDisplayMessagesResponse	9. The Test System sends a GetDisplayMessagesRequest with id <Generated displayMessageId2> requestId <Generated requestId>
11. The Charging Station sends a NotifyDisplayMessagesRequest	12. The Test System responds with a NotifyDisplayMessagesResponse .
13. Execute Reusable State <i>StopAuthorized</i>	
14. Execute Reusable State <i>EVConnectedPostSession</i>	
15. Execute Reusable State <i>EVDisconnected</i>	
16. Execute Reusable State <i>ParkingBayUnoccupied</i>	

Main (Test scenario)	
<u>Note(s):</u> - The display messages are not displayed anymore	
18. The Charging Station responds with a GetDisplayMessagesResponse	17. The Test System sends a GetDisplayMessagesRequest with id <Generated displayMessageId>
20. The Charging Station responds with a GetDisplayMessagesResponse	19. The Test System sends a GetDisplayMessagesRequest with id <Configured displayMessage2Id>

Tool validations
<p>* Step 2: Message SetDisplayMessageResponse - status <i>Accepted</i></p> <p>* Step 4: Message SetDisplayMessageResponse - status <i>Accepted</i></p> <p>* Step 6: Message: GetDisplayMessagesResponse - status <i>Accepted</i></p> <p>* Step 7: Message: NotifyDisplayMessagesRequest - requestId <Generated RequestId> - transactionId <Generated transactionId> - priority <i>InFront</i> - message.content <Configured message></p> <p>* Step 10: Message: GetDisplayMessagesResponse - status <i>Accepted</i></p> <p>* Step 11: Message: NotifyDisplayMessagesRequest - requestId <Generated RequestId> - transactionId <Generated transactionId> - priority <i>InFront</i> - message.content <Configured message with a " 2" extended to it.></p> <p>* Step 18: Message: GetDisplayMessagesResponse - status <i>Unknown</i></p> <p>* Step 20: Message: GetDisplayMessagesResponse - status <i>Unknown</i></p>
Post scenario validations: - N/a

TC_O_32_CS: Set Display Message - Specific transaction - Second Alwaysfront priority

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
Reusable State: State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetDisplayMessageResponse	1. The Test System sends a SetDisplayMessageRequest with message.id <Generated displayMessageId> message.transactionId <Received transactionId> AND message.priority AlwaysFront AND message.display <Configured display component variable> (Omitted when left empty)
<u>Note(s):</u> - Display message <Generated displayMessageId> is shown	
4. The Charging Station responds with a SetDisplayMessageResponse	3. The Test System sends a SetDisplayMessageRequest with message.id <Configured displayMessage2Id> message.transactionId <Received transactionId> AND message.priority AlwaysFront AND message.display <Configured display component variable> (Omitted when left empty)
<u>Note(s):</u> - Display message <Generated displayMessage1Id> is not displayed anymore - Display message <Generated displayMessage2Id> is shown	
6. The Charging Station responds with a GetDisplayMessagesResponse	5. The Test System sends a GetDisplayMessagesRequest with id <Generated displayMessageId>
8. The Charging Station responds with a GetDisplayMessagesResponse	7. The Test System sends a GetDisplayMessagesRequest with id <Generated displayMessageId2> requestId <Generated requestId>
9. The Charging Station sends a NotifyDisplayMessagesRequest	10. The Test System responds with a NotifyDisplayMessagesResponse .
11. Execute Reusable State <i>StopAuthorized</i>	
12. Execute Reusable State <i>EVConnectedPostSession</i>	

Main (Test scenario)	
13. Execute Reusable State <i>EVDisconnected</i>	
14. Execute Reusable State <i>ParkingBayUnoccupied</i>	
Note(s): - Display message <Generated displayMessage2Id> is not displayed anymore	
16. The Charging Station responds with a GetDisplayMessagesResponse	15. The Test System sends a GetDisplayMessagesRequest with id <Generated displayMessageId2>

Tool validations
<p>* Step 2: Message SetDisplayMessageResponse - status <i>Accepted</i></p> <p>* Step 4: Message SetDisplayMessageResponse - status <i>Accepted</i></p> <p>* Step 6: Message GetDisplayMessagesResponse - status <i>Unknown</i></p> <p>* Step 8: Message GetDisplayMessagesResponse - status <i>Accepted</i></p> <p>* Step 9: Message: NotifyDisplayMessagesRequest - requestId <Generated RequestId> - transactionId <Generated transactionId> - priority <i>AlwaysFront</i> - message.content <Configured message with a "2" extended to it.></p> <p>* Step 16: Message: GetDisplayMessagesResponse - status <i>Unknown</i></p> <p>Post scenario validations: - N/a</p>

TC_O_33_CS: Get a Specific Display Message - No DisplayMessages configured

Test case name	Get a Specific Display Message - No DisplayMessages configured
Test case Id	TC_O_33_CS
Use case Id(s)	004
Requirement(s)	004_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

Reusable State:

N/a

Main (Test scenario)

Charging Station	CSMS
2. The Charging Station responds with a GetDisplayMessagesResponse	1. The Test System sends a GetDisplayMessagesRequest with id <Generated displayMessageId>

Tool validations

* Step 2:

Message **GetDisplayMessagesResponse**- **status** *Unknown*

Post scenario validations:

- N/a

TC_O_34_CS: Get a Specific Display Message - Known Id, but not matching State

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)	004
Requirement(s)	004_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:

[SetDisplayMessage](#) with state <Configured Message State>

Reusable State:

N/a

Main (Test scenario)

Charging Station	CSMS
2. The Charging Station responds with a GetDisplayMessagesResponse	1. The Test System sends a GetDisplayMessagesRequest with id <Generated displayMessageId> AND state <Configured Message State 2>

Tool validations

* Step 2:

Message **GetDisplayMessagesResponse**- **status** *Unknown*

Post scenario validations:

- N/a

TC_O_35_CS: Get a Specific Display Message - Known Id, but not matching Priority

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)	004
Requirement(s)	004_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: SetDisplayMessage
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a GetDisplayMessagesResponse	1. The Test System sends a GetDisplayMessagesRequest with id <Generated displayMessageId> AND state <Configured Message Priority 2>

Tool validations
* Step 2: Message GetDisplayMessagesResponse - status <i>Unknown</i>
Post scenario validations: - N/a

TC_O_36_CS: Set Display Message - State Charging

Test case name	Set Display Message - State Charging
Test case Id	TC_O_36_CS
Use case Id(s)	001
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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetDisplayMessageResponse	1. The Test System sends a SetDisplayMessageRequest with message.id <Generated displayMessageId> message.priority <Configured Priority> message.state Charging message.display <Configured display component variable> (Omitted when left empty)
Note(s): The display message should NOT be displayed.	
3. Execute Reusable State <i>ParkingBayOccupied</i>	
4. Execute Reusable State <i>Authorized</i>	
5. Execute Reusable State <i>EVConnectedPreSession</i>	
6. Execute Reusable State <i>EnergyTransferStarted</i>	
Note(s): The display message should be displayed.	
7. Execute Reusable State <i>StopAuthorized</i>	
8. Execute Reusable State <i>EVConnectedPostSession</i>	
9. Execute Reusable State <i>EVDisconnected</i>	
10. Execute Reusable State <i>ParkingBayUnoccupied</i>	
Note(s): The display message should NOT be displayed.	
12. The Charging Station responds with a GetDisplayMessagesResponse	11. The Test System sends a GetDisplayMessagesRequest with id <Generated displayMessageId> requestId <Generated requestId>
13. The Charging Station sends a NotifyDisplayMessagesRequest	14. The Test System responds with a NotifyDisplayMessagesResponse .
Note(s): If <i>tbc</i> is True at Step 13 then step 13 and 14 will be repeated	

Tool validations
<div>* Step 2: Message SetDisplayMessageResponse - status <i>Accepted</i></div> <div>* Step 12: Message GetDisplayMessagesResponse - status <i>Accepted</i></div> <div>* Step 13: Message NotifyDisplayMessagesRequest - requestId <i><Generated requestId></i> - state <i>Charging</i></div>
<div>Post scenario validations: - N/a</div>

TC_O_37_CS: Set Display Message - State Idle

Test case name	Set Display Message - State Idle
Test case Id	TC_O_37_CS
Use case Id(s)	001
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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetDisplayMessageResponse	1. The Test System sends a SetDisplayMessageRequest with message.id <Generated displayMessageId> message.priority <Configured Priority> message.state Idle message.display <Configured display component variable> (Omitted when left empty)
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 Test System sends a GetDisplayMessagesRequest with id <Generated displayMessageId> requestId <Generated requestId>
13. The Charging Station sends a NotifyDisplayMessagesRequest	14. The Test System responds with a NotifyDisplayMessagesResponse .
Note(s): If tbc is True at Step 13 then step 13 and 14 will be repeated	

Tool validations
<div>* Step 2: Message SetDisplayMessageResponse - status <i>Accepted</i></div> <div>* Step 12: Message GetDisplayMessagesResponse - status <i>Accepted</i></div> <div>* Step 13: Message NotifyDisplayMessagesRequest - requestId <i><Generated requestId></i> - state <i>Idle</i></div>
Post scenario validations: - N/a

TC_O_38_CS: Set Display Message - State Unavailable

Test case name	Set Display Message - State Unavailable
Test case Id	TC_O_38_CS
Use case Id(s)	001
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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetDisplayMessageResponse	1. The Test System sends a SetDisplayMessageRequest with message.id <Generated displayMessageId> message.priority <Configured Priority> message.state Unavailable message.display <Configured display component variable> (Omitted when left empty)
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 Test System 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 Test System responds accordingly.
Note(s): The display message should NOT be displayed.	
9. The Charging Station responds with a GetDisplayMessagesResponse	8. The Test System sends a GetDisplayMessagesRequest with id <Generated displayMessageId> requestId <Generated requestId>
10. The Charging Station sends a NotifyDisplayMessagesRequest	11. The Test System responds with a NotifyDisplayMessagesResponse .
Note(s): If tbc is True at Step 10 then step 10 and 11 will be repeated	

Tool validations
<p>* Step 2:</p> <p>Message SetDisplayMessageResponse</p> <ul style="list-style-type: none"> - status <i>Accepted</i> <p>* Step 5:</p> <p>Message ChangeAvailabilityResponse</p> <ul style="list-style-type: none"> - status <i>Accepted</i> <p>* Step 6:</p> <p>Message: StatusNotificationRequest</p> <ul style="list-style-type: none"> - connectorStatus <i>Available</i> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none"> - eventData[0].trigger <i>Delta</i> - eventData[0].actualValue <i>"Available"</i> - eventData[0].component.name <i>"ChargingStation" / EVSE / Connector</i> - eventData[0].variable.name <i>"AvailabilityState"</i> <p>* Step 9:</p> <p>Message GetDisplayMessagesResponse</p> <ul style="list-style-type: none"> - status <i>Accepted</i> <p>* Step 10:</p> <p>Message NotifyDisplayMessagesRequest</p> <ul style="list-style-type: none"> - requestId <i><Generated requestId></i> - state <i>Unavailable</i>
<p>Post scenario validations:</p> <ul style="list-style-type: none"> - N/a

TC_O_39_CS: Set Display Message - State Faulted

Test case name	Set Display Message - State Faulted
Test case Id	TC_O_39_CS
Use case Id(s)	001
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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetDisplayMessageResponse	1. The Test System sends a SetDisplayMessageRequest with message.id <Generated displayMessageId> message.priority <Configured Priority> message.state Faulted message.message.content <Message indicating the Charging Station is in a Faulted state> message.display <Configured display component variable> (Omitted when left empty)
Note(s): The display message should NOT be displayed.	
Manual Action: Set the Charging Station to state Faulted.	
3. The Charging Station notifies the CSMS about the status change of the Charging Station.	4. The Test System responds accordingly.
Note(s): - Step 3/4 are used to detect if the Charging Station status has changed. - The display message should be displayed now.	
Manual Action: Set the Charging Station back to state Available.	
5. The Charging Station notifies the CSMS about the status change of the Charging Station.	6. The Test System responds accordingly.
Note(s): - Step 5/6 are used to detect if the Charging Station status has changed. - The display message should NOT be displayed anymore.	
8. The Charging Station responds with a GetDisplayMessagesResponse	7. The Test System sends a GetDisplayMessagesRequest with id <Generated displayMessageId> requestId <Generated requestId>
9. The Charging Station sends a NotifyDisplayMessagesRequest	10. The Test System responds with a NotifyDisplayMessagesResponse .
Note(s): If tbc is True at Step 9 then step 9 and 10 will be repeated	

Tool validations
<p>* Step 2:</p> <p>Message SetDisplayMessageResponse</p> <ul style="list-style-type: none"> - status <i>Accepted</i> <p>* Step 3:</p> <p>At least one of the following messages must be sent:</p> <p>Message: StatusNotificationRequest</p> <ul style="list-style-type: none"> - connectorStatus <i>Faulted</i> or <i>Unavailable</i> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none"> - eventData[0].trigger must be <i>Delta</i> - eventData[0].actualValue must be <i>Faulted</i> or <i>Unavailable</i> - eventData[0].component.name must be <i>Connector</i> - eventData[0].variable.name must be <i>AvailabilityState</i> - evse.id <i><not omitted></i> - connector.id <i><not omitted></i> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none"> - eventData[0].trigger must be <i>Delta</i> - eventData[0].actualValue must be <i>Faulted</i> - eventData[0].component.name must be <i>ChargingStation</i> - eventData[0].variable.name must be <i>AvailabilityState</i> - evse.id <i><omitted></i> - connector.id <i><omitted></i> <p>* Step 5:</p> <p>At least one of the following messages must be sent:</p> <p>Message: StatusNotificationRequest</p> <ul style="list-style-type: none"> - connectorStatus <i>Available</i> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none"> - eventData[0].trigger must be <i>Delta</i> - eventData[0].actualValue must be <i>Available</i> - eventData[0].component.name must be <i>Connector</i> - eventData[0].variable.name must be <i>AvailabilityState</i> - evse.id <i><not omitted></i> - connector.id <i><not omitted></i> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none"> - eventData[0].trigger must be <i>Delta</i> - eventData[0].actualValue must be <i>Available</i> - eventData[0].component.name must be <i>ChargingStation</i> - eventData[0].variable.name must be <i>AvailabilityState</i> <p>* Step 8:</p> <p>Message GetDisplayMessagesResponse</p> <ul style="list-style-type: none"> - status <i>Accepted</i> <p>* Step 9:</p> <p>Message NotifyDisplayMessagesRequest</p> <ul style="list-style-type: none"> - requestId <i><Generated requestId></i> - state <i>Faulted</i>
<p>Post scenario validations:</p> <ul style="list-style-type: none"> - N/a

TC_O_100_CS: Set Display Message - unsupported language

Test case name	Set Display Message - unsupported language
Test case Id	TC_O_100_CS
Use case Id(s)	O01
Requirement(s)	O01.FR.18
System under test	Charging Station
Description	This test case describes how the CSMS can be requested to send a 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 language being set is not supported.
Prerequisite(s)	N/a

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetDisplayMessageResponse	1. The Test System sends a SetDisplayMessageRequest with message.id <Generated displayMessageId> message.message[0].language <Configured supported languages> message.message[0].content contentSupported message.messageExtra[0].language <configured not supported languages> message.messageExtra[0].content contentUnSupported

Tool validations
* Step 2: Message SetDisplayMessageResponse - status must be <i>LanguageNotSupported</i>
Post scenario validations: - N/a

TC_O_101_CS: Set Display Message - Language preference of the EV Driver

Test case name	Set Display Message - Language preference of the EV Driver
Test case Id	TC_O_101_CS
Use case Id(s)	O01
Requirement(s)	C01.FR.08, O01.FR.08, O01.FR.09
System under test	Charging Station
Description	This test case describes how the CSMS can be requested to send a 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 messages in the preferred language of the EV driver.
Prerequisite(s)	N/a

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
Manual Action: Present idToken.	
<p>1. The Charging Station sends an AuthorizeRequest</p> <p><u>Note 1:</u> This step needs to be executed: - unless (AuthEnabled is implemented with mutability <i>ReadOnly</i> 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 Test System) OR - the idToken is cached. <u>Note 2:</u> Alternatively, EV Driver may select <Configured language1> using button or display on charging station.</p>	<p>2. The Test System responds with an AuthorizeResponse with idTokenInfo.status <i>Accepted</i> idTokenInfo.language1 <Configured language1></p>
<p><u>Note(s):</u> - The Charging Station will use the language preference of the EV Driver to display messages.</p>	
<p>4. The Charging Station responds with a SetDisplayMessageResponse</p>	<p>3. The Test System sends a SetDisplayMessageRequest with message.id <Generated displayMessageId> message.message[0].language <Configured supported languages> message.message[0].content <i>contentSupported</i> message.messageExtra[0].language <Configured language1> message.messageExtra[0].content <i>ContentInCustomerLanguage1</i></p>

Tool validations
<p>* Step 1: Message: AuthorizeRequest - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type></p> <p>* Step 4: Visual inspection Message: SetDisplayMessageResponse - The Message <i>ContentInCustomerLanguage1</i> is shown on display.</p>

Tool validations
Post scenario validations: - After step 2, the Charging Station will show messages with language preference (<Configured language1>) of the EV Driver, including the SetDisplayMessage of step 3.

P Data Transfer

TC_P_01_CS: Data Transfer to the Charging Station - Rejected / Unknown VendorId / Unknown MessageId

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 Test System sends a DataTransferRequest with vendorId <i>org.openchargealliance.Test System</i> messageId <i><Configured messageId></i>

Tool validations
* Step 2: Message: DataTransferResponse - status must be <i>UnknownVendorId</i> OR <i>UnknownMessageId</i> OR <i>Rejected</i> (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

TC_P_03_CS: CustomData - Receive custom data

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with SetVariablesResponse	1. Test System sends SetVariablesRequest with: - variable.name = "OfflineThreshold" - component.name = "OCPPCommCtrlr" - attributeValue = "200" - attributeType is Actual
4. The Charging Station responds with GetVariablesResponse	3. Test System sends GetVariablesRequest with: - variable.name = "OfflineThreshold" - component.name = "OCPPCommCtrlr" - attributeType is Actual

Tool validations
* Step 2: Message: SetVariablesResponse - setVariableResult[0].attributeStatus <i>Accepted</i> * Step 4: Message: GetVariablesResponse - getVariableResult[0].attributeStatus <i>Accepted</i> - getVariableResult[0].attributeType <i>Actual</i> or omitted - getVariableResult[0].attributeValue <i>200</i>
Post scenario validations: - N/a

TC_P_04_CS: Able to receive customData - ChargingProfile

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetChargingProfileResponse	<p>1. The Test System sends a SetChargingProfileRequest with evseld <Configured evseld> 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></p>

Tool validations
* Step 2: Message SetChargingProfileResponse - status Accepted
Post scenario validations: - N/a

Q Bidirectional Power Transfer

TC_Q_100_CS: V2X Authorisation - ISO15118-20 - V2X Tx Measurands defined

Test case name	V2X Authorisation - V2X Tx Measurands defined
Test case Id	TC_Q_100_CS
Use case Id(s)	Q01
Requirement(s)	Q01.FR.33, Q01.FR.34, Q01.FR.35
System under test	Charging Station
Description	Authorization of an EV by the CSMS to start a V2X power transfer loop.
Purpose	To verify if the Charging Station is able to send the configured V2X measurands.
Prerequisite(s)	<ul style="list-style-type: none"> - EV supports V2X power transfer loop - EV and charging station support ISO15118-20 or CHAdeMO - V2XChargingCtrlr.SupportedOperationModes contains <i>CentralSetpoint</i>

Before (Preparations)

Configuration State:

ISO15118Ctrlr.Enabled is *true* or absentV2XChargingCtrlr.Enabled = *true*

V2XChargingCtrlr

- Enabled is *true*
- TxStartedMeasurands[CentralSetpoint] is <Configured V2XTxStartedMeasurands>
- TxEndedMeasurands[CentralSetpoint] is <Configured V2XTxUpdatedMeasurands>
- TxUpdatedMeasurands[CentralSetpoint] is <Configured V2XTxEndedMeasurands>
- TxUpdatedInterval is 5
- TxEndedInterval is 0

SampledDataCtrlr

- TxStartedMeasurands is [Energy.Active.Import.Register]
- TxUpdatedMeasurands is [Energy.Active.Import.Register]
- TxEndedMeasurands is [Energy.Active.Import.Register]
- TxUpdatedInterval is 5
- TxEndedInterval is 0

Memory State:

N/a

Reusable State(s):

IF <Configured EVProtocol> is CHAdeMO THEN

State is *Authorized*State is *EVConnectedPreSession*

ELSE IF <Configured EVProtocol> is ISO15118 THEN

State is *Iso1511820V2xControlBpt*State is *EVConnectedPreSession*State is *Authorized15118* (for 15118-20)

END IF

Main (Test scenario)

Charging Station	CSMS
------------------	------

Main (Test scenario)	
1. The Charging Station sends a <code>NotifyEVChargingNeedsRequest</code> <u>Note(s):</u> - This step is applicable if <Configured EVProtocol> is ISO15118 and is captured by Authorized15118 (for 15118-20), but validated on testcase level.	2. Test System responds with a <code>NotifyEVChargingNeedsResponse</code> with <code>status</code> <i>Accepted</i>
4. The Charging Station responds with a <code>SetChargingProfileResponse</code>	3. The Test System sends a <code>SetChargingProfileRequest</code> with <code>evseld</code> <Configured evseld> <code>chargingProfile</code> - <code>transactionId</code> <transactionId> - <code>chargingProfilePurpose</code> TxProfile - <code>chargingSchedule[0].chargingSchedulePeriod[0].operationMode</code> CentralSetpoint
5. The Charging station sends a <code>NotifyEVChargingScheduleRequest</code> <u>Note(s):</u> - This step is applicable if <Configured EVProtocol> is ISO15118	6. The Test System responds with a <code>NotifyEVChargingScheduleResponse</code> with <code>status</code> <i>Accepted</i>
7. Execute Reusable State EnergyTransferStarted	
8. The Charging Station sends a <code>TransactionEventRequest</code> <u>Note(s):</u> - This step is captured by the reusable states, but validated on testcase level.	9. The Test System responds with a <code>TransactionEventResponse</code>
<i>The SampledDataCtrlr.TxUpdatedInterval expires</i>	
10. The Charging Station sends a <code>TransactionEventRequest</code>	11. The Test System responds with a <code>TransactionEventResponse</code>
<u>Note(s):</u> The Charging Station receives a <code>SessionStopReq(Terminate)</code> message from the EV.	
IF <Configured EVProtocol> is CHAdeMO THEN	
12a. Execute Reusable State StopAuthorized (local)	
ELSE IF <Configured EVProtocol> is ISO15118 THEN	
12b. Execute Reusable State EnergyTransferSuspended	
13. Execute Reusable State EVDisconnected	
14. Execute Reusable State ParkingBayUnoccupied	
15. The Charging Station sends a <code>TransactionEventRequest</code> <u>Note(s):</u> - This step is captured by the reusable state steps, but an additional validation is described on testcase level.	16. The Test System responds with a <code>TransactionEventResponse</code>

Tool validations
<p>* Step 1:</p> <p>Message NotifyEVChargingNeedsRequest</p> <ul style="list-style-type: none"> - evseld = <Configured evseld> <p>For AC charging station:</p> <ul style="list-style-type: none"> - chargingNeeds.requestedEnergyTransfer AC_BPT - chargingNeeds.availableEnergyTransfer [AC_single_phase, AC_two_phase, AC_three_phase, AC_BPT, AC_BPT_DER, AC_DER] <p>For DC charging station:</p> <ul style="list-style-type: none"> - chargingNeeds.requestedEnergyTransfer DC_BPT - chargingNeeds.availableEnergyTransfer [DC, DC_BPT, DC_ACDP, DC_ACDP_BPT] <ul style="list-style-type: none"> - chargingNeeds.controlMode = <i>DynamicControl</i> - chargingNeeds.v2xChargingParameters.maxChargePower = 10000 (W) - chargingNeeds.v2xChargingParameters.maxDischargePower = 5000 (W) <p>* Step 3:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - eventType must be <i>Started</i> - meterValue[*].sampledValue[*].measurand must contain <Configured V2XTxStartedMeasurands> and <i>Energy.Active.Import.Register</i> <p>* Step 10:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - eventType must be <i>Updated</i> - meterValue[*].sampledValue[*].measurand must contain <Configured V2XTxUpdatedMeasurands> and <i>Energy.Active.Import.Register</i> <p>* Step 15:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - eventType must be <i>Ended</i> - meterValue[*].sampledValue[*].measurand must contain <Configured V2XTxEndedMeasurands> and <i>Energy.Active.Import.Register</i>
<p>Post scenario validations:</p> <p>N/a</p>

TC_Q_101_CS: V2X Authorisation - ISO15118-20 - Processing charging needs

Test case name	V2X Authorisation - ISO15118-20 - Processing charging needs
Test case Id	TC_Q_101_CS
Use case Id(s)	Q01
Requirement(s)	Q01.FR.03, Q01.FR.06, K17.FR.10
System under test	Charging Station
Description	Authorization of an EV by the CSMS to start a V2X power transfer loop.
Purpose	To verify if the Charging Station is able to provide bidirectional charging needs.
Prerequisite(s)	- EV supports V2X power transfer loop - EV and charging station support ISO15118-20

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is Iso1511820V2xControlBpt with controlMode <i>ScheduledControl</i> State is EVConnectedPreSession State is Authorized15118 (for 15118-20)

Main (Test scenario)	
Charging Station	CSMS
<i>Agree on energy transfer mode</i>	
1. The Charging Station sends a NotifyEVChargingNeedsRequest <u>Note(s):</u> - This step is captured by Authorized15118 (for 15118-20), but validated on testcase level.	2. Test System responds with a NotifyEVChargingNeedsResponse with status <i>Accepted</i>
<i>Setting charging profile</i>	
4. The Charging Station responds with a SetChargingProfileResponse	3. The Test System sends a SetChargingProfileRequest with evseld <Configured evseld> chargingProfile - transactionId <transactionId> - chargingProfilePurpose <i>TxProfile</i>
5. The Charging station sends a NotifyEVChargingScheduleRequest	6. The Test System responds with a NotifyEVChargingScheduleResponse with status <i>Accepted</i>
7. Execute Reusable State EnergyTransferStarted	

Tool validations
<p>* Step 1:</p> <p>Message: NotifyEVChargingNeedsRequest</p> <ul style="list-style-type: none">- evseld must be <i><Configured evseld></i>- chargingNeeds.requestedEnergyTransfer must be <i>AC_BPT</i> or <i>DC_BPT</i>- chargingNeeds.availableEnergyTransfer must contain <i>[AC_single_phase, AC_BPT]</i> or <i>[DC, DC_BPT]</i>- chargingNeeds.controlMode must be <i>ScheduledControl</i>- chargingNeeds.v2xChargingParameters.maxChargePower must be <i>10000</i>- chargingNeeds.v2xChargingParameters.maxDischargePower must be <i>5000</i> <p>* Step 4:</p> <p>Message: SetChargingProfileResponse</p> <ul style="list-style-type: none">- status must be <i>Accepted</i> <p>* Step 5:</p> <p>Message: NotifyEVChargingScheduleRequest</p> <ul style="list-style-type: none">- evseld must be <i><Configured evseld></i>
<p>Post scenario validations:</p> <p>N/a</p>

TC_Q_102_CS: V2X Authorisation - ISO15118-20 - Charging only (V2X control) before starting V2X - Allowed Energy Transfer modes omitted

Test case name	V2X Authorisation - ISO15118-20 - Charging only (V2X control) before starting V2X - Allowed Energy Transfer modes omitted
Test case Id	TC_Q_102_CS
Use case Id(s)	Q02
Requirement(s)	Q02.FR.04
System under test	Charging Station
Description	Authorization of an EV by the CSMS to start a V2X power transfer loop.
Purpose	To verify if the Charging Station will enable charging only when CSMS doesn't provide an allowed energy transfer modes.
Prerequisite(s)	- EV supports V2X power transfer loop - EV and charging station support ISO15118-20

Before (Preparations)
Configuration State: - ISO15118Ctrlr.Enabled is <i>true</i> - V2XChargingCtrlr.SupportedEnergyTransferMode contains [AC_single_phase, AC_BPT] or [DC, DC_BPT]
Memory State: N/a
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
Communicate that default charging station energy transfer mode will be used	
1. Execute Reusable State <i>Iso1511820V2xControlBpt</i> with controlMode <i>ScheduledControl</i>	
2. Execute Reusable State <i>EVConnectedPreSession</i>	
3. Execute Reusable State <i>Authorized15118</i> (for 15118-20)	
<u>Note(s):</u> - The Test System omits the <i>allowedEnergyTransfer</i> from the <i>AuthorizeResponse</i> .	
4. The Charging Station sends a NotifyEVChargingNeedsRequest	5. Test System responds with a NotifyEVChargingNeedsResponse with status <i>Accepted</i>
<u>Note(s):</u> - This step is captured by <i>Authorized15118</i> (for 15118-20), but validated on testcase level.	
6. Execute Reusable State <i>EnergyTransferStarted</i>	

Tool validations
<p>* Step 4:</p> <p>Message: NotifyEVChargingNeedsRequest</p> <ul style="list-style-type: none">- evseld must be <i><Configured evseld></i> <p><u>For AC charging station:</u></p> <ul style="list-style-type: none">- chargingNeeds.requestedEnergyTransfer <i>AC_three_phase</i>- chargingNeeds.availableEnergyTransfer <i>[AC_single_phase, AC_two_phase, AC_three_phase, AC_BPT, AC_BPT_DER, AC_DER]</i> <p><u>For DC charging station:</u></p> <ul style="list-style-type: none">- chargingNeeds.requestedEnergyTransfer <i>DC</i>- chargingNeeds.availableEnergyTransfer <i>[DC, DC_BPT, DC_ACDP, DC_ACDP_BPT]</i> <ul style="list-style-type: none">- chargingNeeds.controlMode must be <i>ScheduledControl</i>- chargingNeeds.v2xChargingParameters.maxChargePower must be <i>10000 (W)</i>- chargingNeeds.v2xChargingParameters.maxDischargePower must be <i>5000 (W)</i>
<p>Post scenario validations:</p> <p>N/a</p>

TC_Q_103_CS: V2X Authorisation - ISO15118-20 - Charging needs rejected

Test case name	V2X Authorisation - ISO15118-20 - Charging needs rejected
Test case Id	TC_Q_103_CS
Use case Id(s)	Q01
Requirement(s)	Q01.FR.03, Q01.FR.06
System under test	Charging Station
Description	Authorization of an EV by the CSMS to start a V2X power transfer loop.
Purpose	To verify if the Charging Station is able to end the transaction when the charging needs are rejected and it doesn't support ISO15118ServiceRenegotiationSupport.
Prerequisite(s)	<ul style="list-style-type: none"> - EV supports V2X power transfer loop - EV and charging station support ISO15118-20 - Charging Station doesn't support ISO15118ServiceRenegotiationSupport (component variable <i>ISO15118Ctrlr.ServiceRenegotiationSupport</i> is <i>false</i> or absent) - Charging Station has a TxStartPoint containing one or more of <i>ParkingBayOccupancy</i>, <i>EVConnected</i>, <i>Authorized</i> or <i>PowerPathClosed</i> - EV selects a BPT service

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>Iso1511820V2xControlBpt</i> State is <i>EVConnectedPreSession</i> State is <i>Authorized15118</i> (for 15118-20)

Main (Test scenario)	
Charging Station	CSMS
1. The Charging Station sends a NotifyEVChargingNeedsRequest <u>Note(s):</u> - This step is captured by <i>Authorized15118</i> (for 15118-20), but validated on testcase level.	2. The Test System responds with a NotifyEVChargingNeedsResponse with - status <i>Rejected</i>
3. The Charging Station sends a TransactionEventRequest	4. The Test System responds with a TransactionEventResponse

Tool validations
<p>* Step 1:</p> <p>Message: NotifyEVChargingNeedsRequest</p> <ul style="list-style-type: none">- evseld must be <i><Configured evseld></i> <p><u>For AC charging station:</u></p> <ul style="list-style-type: none">- chargingNeeds.requestedEnergyTransfer <i>AC_BPT</i>- chargingNeeds.availableEnergyTransfer one or more of <i>[AC_single_phase, AC_two_phase, AC_three_phase, AC_BPT, AC_BPT_DER, AC_DER]</i> <p><u>For DC charging station:</u></p> <ul style="list-style-type: none">- chargingNeeds.requestedEnergyTransfer <i>DC_BPT</i>- chargingNeeds.availableEnergyTransfer one of more of <i>[DC, DC_BPT, DC_ACDP, DC_ACDP_BPT]</i> <ul style="list-style-type: none">- chargingNeeds.controlMode must be <i><not omitted></i>- chargingNeeds.v2xChargingParameters.maxChargePower must be <i><not omitted></i>- chargingNeeds.v2xChargingParameters.maxDischargePower must be <i><not omitted></i> <p>* Step 3:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none">- eventType must be <i>Ended</i>- transactionInfo.transactionId must be <i><transactionId></i>- triggerReason must be <i>AbnormalCondition</i>- transactionInfo.transactionId must be <i><transactionId></i>- transactionInfo.stoppedReason must be <i>ReqEnergyTransferRejected</i>
<p>Post scenario validations:</p> <p>N/a</p>

TC_Q_104_CS: V2X Authorisation - ISO15118-20 - Scheduled Control

Test case name	V2X Authorisation - ISO15118-20 - Scheduled Control
Test case Id	TC_Q_104_CS
Use case Id(s)	Q01, Q02
Requirement(s)	Q01.FR.03, Q02.FR.05, Q02.FR.07
System under test	Charging Station
Description	Authorization of an EV by the CSMS to start a V2X power transfer loop.
Purpose	To verify if the Charging Station is able to support ISO15118-20 Schedule Control.
Prerequisite(s)	- EV supports V2X power transfer loop - EV and charging station support ISO15118-20

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is Iso1511820V2xControlBpt with controlMode <i>ScheduledControl</i> State is EVConnectedPreSession State is Authorized15118 (for 15118-20)

Main (Test scenario)	
Charging Station	CSMS
<p>1. The Charging Station sends a NotifyEVChargingNeedsRequest</p> <p><u>Note(s):</u> - This step is captured by Authorized15118 (for 15118-20), but validated on testcase level.</p>	<p>2. The Test System responds with a NotifyEVChargingNeedsResponse with</p> <ul style="list-style-type: none">- status <i>Accepted</i>
Setting charging profile	
<p>4. The Charging Station responds with a SetChargingProfileResponse</p>	<p>3. The Test System sends a SetChargingProfileRequest with</p> <p>evseld <i><Configured evseld></i></p> <p>chargingProfile</p> <ul style="list-style-type: none">- transactionId <i><transactionId></i>- chargingProfilePurpose <i>TxProfile</i>- chargingProfile.chargingSchedule[0].chargingRateUnit <i>W</i>chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].startPeriod <i>0</i>chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].limit <i>1000</i>chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].operationMode <i>ChargingOnly</i>chargingProfile.chargingSchedule[0].chargingSchedulePeriod[1].startPeriod <i>300</i>chargingProfile.chargingSchedule[0].chargingSchedulePeriod[1].limit <i>2000</i>chargingProfile.chargingSchedule[0].chargingSchedulePeriod[1].operationMode <i>ChargingOnly</i>
<p>5. The Charging station sends a NotifyEVChargingScheduleRequest</p>	<p>6. The Test System responds with a NotifyEVChargingScheduleResponse with</p> <ul style="list-style-type: none">status <i>Accepted</i>
7. Execute Reusable State EnergyTransferStarted	

Tool validations
<p>* Step 1:</p> <p>Message: NotifyEVChargingNeedsRequest</p> <ul style="list-style-type: none">- evseld must be <i><Configured evseld></i> <p><u>For AC charging station:</u></p> <ul style="list-style-type: none">- chargingNeeds.requestedEnergyTransfer <i>AC_BPT</i>- chargingNeeds.availableEnergyTransfer <i>[AC_single_phase, AC_two_phase, AC_three_phase, AC_BPT, AC_BPT_DER, AC_DER]</i> <p><u>For DC charging station:</u></p> <ul style="list-style-type: none">- chargingNeeds.requestedEnergyTransfer <i>DC_BPT</i>- chargingNeeds.availableEnergyTransfer <i>[DC, DC_BPT, DC_ACDP, DC_ACDP_BPT]</i> <ul style="list-style-type: none">- chargingNeeds.controlMode must be <i>ScheduledControl</i>- chargingNeeds.v2xChargingParameters.maxChargePower must be <i>10000 (W)</i>- chargingNeeds.v2xChargingParameters.maxDischargePower must be <i>5000 (W)</i> <p>* Step 4:</p> <p>Message: SetChargingProfileResponse</p> <ul style="list-style-type: none">- status must be <i>Accepted</i> <p>* Step 5:</p> <p>Message: NotifyEVChargingScheduleRequest</p> <ul style="list-style-type: none">- evseld must be <i><Configured evseld></i>- chargingSchedule.chargingSchedulePeriod[0].startPeriod <i>0</i>- chargingSchedule.chargingSchedulePeriod[0].limit <i><= 1000</i>- chargingSchedule.chargingSchedulePeriod[1].startPeriod <i>300</i>- chargingSchedule.chargingSchedulePeriod[1].limit <i><= 2000</i> <p>Post scenario validations:</p> <p>N/a</p>

TC_Q_107_CS: V2X Authorisation - ISO15118-20 - Charging only (V2X control) before starting V2X

Test case name	V2X Authorisation - ISO15118-20 - Charging only (V2X control) before starting V2X
Test case Id	TC_Q_107_CS
Use case Id(s)	Q02
Requirement(s)	Q01.FR.02, Q01.FR.03, Q02.FR.04, Q02.FR.06, Q02.FR.07 Q02.FR.04, Q02.FR.06, Q02.FR.07
System under test	Charging Station
Description	To describe starting a transaction in ChargingOnly mode before getting approval to do V2X.
Purpose	To verify if the Charging station uses Charging Only when CSMS doesn't allow V2X.
Prerequisite(s)	- EV and charging station support ISO15118-20 - Charging Station (is configured to) always start in charging only mode, even when EV requests bidirectional.

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State: State is Iso1511820V2xControlBpt State is EVConnectedPreSession State is Authorized15118 (for 15118-20)

Main (Test scenario)	
Charging Station	CSMS
1. The Charging Station sends a NotifyEVChargingNeedsRequest <u>Note(s):</u> - This step is captured by Authorized15118 (for 15118-20), but validated on testcase level.	2. The Test System responds with a NotifyEVChargingNeedsResponse with - status <i>Accepted</i>
Setting <i>ChargingOnly</i> charging profile	
3. The Charging Station responds with a SetChargingProfileResponse	4. The Test System sends a SetChargingProfileRequest with evseld <i><Configured evseld></i> chargingProfile.chargingProfilePurpose <i>TxProfile</i> chargingProfile.chargingProfileKind <i>Dynamic</i> chargingProfile.transactionId <i><transactionId></i> chargingProfile.chargingSchedule[0].chargingRateUnit <i><Configured chargingRateUnit></i> chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].startPeriod <i>0</i> chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].limit <i>6 * <limit multiplier></i> Note: Check Determine Charging Profile Limit Multiplier for <i><limit multiplier></i> chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].operationMode <i>ChargingOnly</i>
6. The Charging Station responds with a NotifyAllowedEnergyTransferResponse	5. The Test System sends a NotifyAllowedEnergyTransferRequest with allowedEnergyTransfer <i>[AC_single_phase, AC_BPT, DC, DC_BPT]</i>
7. The Charging Station sends a NotifyEVChargingNeedsRequest	8. The Test System responds with a NotifyEVChargingNeedsResponse with - status <i>Accepted</i>

Tool validations
<p>* Step 1:</p> <p>Message: NotifyEVChargingNeedsRequest</p> <ul style="list-style-type: none">- evseld must be <i><Configured evseld></i>- chargingNeeds.requestedEnergyTransfer must be one of <i>[AC_single_phase, AC_two_phase, AC_three_phase, DC, DC_ACDP, WPT]</i>- chargingNeeds.availableEnergyTransfer must be <i><not omitted></i> <p>* Step 3:</p> <p>Message: SetChargingProfileResponse</p> <ul style="list-style-type: none">- status must be <i>Accepted</i> <p>* Step 6:</p> <p>Message: NotifyAllowedEnergyTransferResponse</p> <ul style="list-style-type: none">- status must be <i>Accepted</i> <p>* Step 7:</p> <p>Message: NotifyEVChargingNeedsRequest</p> <ul style="list-style-type: none">- evseld must be <i><Configured evseld></i>- chargingNeeds.requestedEnergyTransfer must be one of <i>[AC_BPT, DC_BPT]</i>- chargingNeeds.availableEnergyTransfer must be <i><not omitted></i>
<p>Post scenario validations:</p> <p>N/a</p>

TC_Q_109_CS: Central dynamic schedule control with setpoint - push

Test case name	Central dynamic schedule control with setpoint - push
Test case Id	TC_Q_109_CS
Use case Id(s)	Q04, K28
Requirement(s)	K28.FR.05, K28.FR.06, K28.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 support DynamicControl charging profiles with the CSMS pushing changes to the CS.
Prerequisite(s)	- Charging station supports <i>SupportsDynamicProfiles</i> . - EV and charging station support ISO15118-20 or CHAdeMO

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 Test System sends a SetChargingProfileRequest with chargingProfile.id 123 chargingProfile.chargingProfilePurpose <i>ChargingStationMaxProfile</i> chargingProfile.chargingProfileKind <i>Dynamic</i> chargingProfile.chargingSchedule[0].chargingRateUnit <Configured chargingRateUnit> chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].startPeriod 0 chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].limit 8 * <limit multiplier> <i>Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></i>
4. The Charging Station responds with a GetChargingProfilesResponse	3. The Test System sends a GetChargingProfilesRequest with chargingProfile.chargingProfileId 123 requestId <Generated requestId>
5. The Charging Station sends a ReportChargingProfilesRequest	6. The Test System responds with a ReportChargingProfilesResponse
Note(s): - If tbc is True at Step 5 then step 5 and 6 will be repeated	
8 The Charging Station responds with a UpdateDynamicScheduleResponse	7 The Test System sends a UpdateDynamicScheduleRequest with chargingProfileId 123 scheduleUpdate.limit 8 * <limit multiplier> <i>Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></i>
10. The Charging Station responds with a GetChargingProfilesResponse	9. The Test System sends a GetChargingProfilesRequest with chargingProfile.chargingProfileId 123 requestId <Generated requestId>

Main (Test scenario)	
11. The Charging Station sends a ReportChargingProfilesRequest	12. The Test System responds with a ReportChargingProfilesResponse
<u>Note(s):</u> - If tbc is True at Step 11 then step 11 and 12 will be repeated	
Tool validations	
* Step 2: Message: SetChargingProfileResponse - status must be <i>Accepted</i> * Step 4: Message: GetChargingProfilesResponse - status must be <i>Accepted</i> * Step 5: Note: The date/time when sending the SetChargingProfileRequest in step 1 is called x : Message: ReportChargingProfilesRequest - chargingProfile.dynUpdateTime must be <x> * Step 8: Message: UpdateDynamicScheduleResponse - status must be <i>Accepted</i> * Step 10: Message: GetChargingProfilesResponse - status must be <i>Accepted</i> * Step 11: Note: The date/time when sending the UpdateDynamicScheduleRequest in step 7 is called y : Message: ReportChargingProfilesRequest - chargingProfile.dynUpdateTime must be <y>	
Post scenario validations: N/a	

TC_Q_110_CS: Central V2X control with dynamic CSMS setpoint - pull

Test case name	Central V2X control with dynamic CSMS setpoint - pull
Test case Id	TC_Q_110_CS
Use case Id(s)	Q04, K28
Requirement(s)	K28.FR.08, K28.FR.09, K28.FR.10
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 support pulling DynamicControl charging profiles from the CSMS.
Prerequisite(s)	- Charging station has SmartChargingCtrlr.SupportsFeature[DynamicProfiles] = true - EV and charging station support ISO15118-20 or CHAdeMO

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): IF <Configured EVProtocol> is CHAdeMO THEN State is Authorized State is EVConnectedPreSession ELSE IF <Configured EVProtocol> is ISO15118 THEN State is Iso1511820V2xControlBpt State is EVConnectedPreSession State is Authorized15118 (for 15118-20) END IF State is EnergyTransferStarted

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetChargingProfileResponse	1. The Test System sends a SetChargingProfileRequest with chargingProfile.id 123 chargingProfile.chargingProfilePurpose <i>TxDefaultProfile</i> chargingProfile.chargingProfileKind <i>Dynamic</i> chargingProfile.dynUpdateTime <i><Current dateTime></i> chargingProfile.dynUpdateInterval 30 chargingProfile.chargingSchedule[0].chargingRateUnit <i><Configured chargingRateUnit></i> chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].startPeriod 0 chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].limit 8 * <i><limit multiplier></i> <i>Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></i>
<u>Note(s)</u> : After about 30 seconds...	
3. The Charging Station sends a PullDynamicScheduleUpdateRequest	4. The Test System responds with a PullDynamicScheduleUpdateResponse with status <i>Accepted</i> scheduleUpdate.limit 6 * <i><limit multiplier></i> <i>Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></i>

Main (Test scenario)	
6. The Charging Station responds with a GetChargingProfilesResponse	5. The Test System sends a GetChargingProfilesRequest with chargingProfile.chargingProfileId 123 requestId <Generated requestId>
7. The Charging Station sends a ReportChargingProfilesRequest	8. The Test System responds with a ReportChargingProfilesResponse
Note(s): - If tbc is True at Step 7 then step 7 and 8 will be repeated	

Tool validations
<p>* Step 2: Message: SetChargingProfileResponse - status must be <i>Accepted</i></p> <p>* Step 3: Message: PullDynamicScheduleUpdateRequest - chargingProfileId must be 123</p> <p>* Step 6: Message: GetChargingProfilesResponse - status must be <i>Accepted</i></p> <p>* Step 7: Message: ReportChargingProfilesRequest Note: The date/time when sending the PullDynamicScheduleUpdateResponse in step 4 is called x: - chargingProfile.dynUpdateTime must be <x> (+/- Max Time Deviation) - chargingProfile.chargingSchedule.id must be 123 - chargingProfile.chargingSchedule.chargingSchedulePeriod[0].limit must be 6 * <limit multiplier> Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></p> <p>Post scenario validations: N/a</p>

TC_Q_111_CS: External V2X control - with a charging profile from CSMS - setpoint

Test case name	External V2X control - with a charging profile from CSMS - setpoint
Test case Id	TC_Q_111_CS
Use case Id(s)	Q05
Requirement(s)	Q05.FR.01, Q05.FR.03, Q05.FR.05
System under test	Charging Station
Description	CSMS explicitly allows an External System to control the charge and discharge behaviour of an EV for a certain period of time.
Purpose	To verify if the Charging Station is able to inform the CSMS when it's capacity is being throttled by a locally connected EMS.
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station is connected locally to a system which can apply charging limit constraints. - SmartChargingCtrlr.SupportsFeature[DynamicProfiles] is <i>true</i>

Before (Preparations)
Configuration State: <ul style="list-style-type: none"> - SmartChargingCtrlr.ExternalControlSignalsEnabled is <i>true</i> - SmartChargingCtrlr.ExternalConstraintsProfileDisallowed is <i>true</i>
Memory State: N/a
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action:</u> Use external connected system to set setpoint to $-6 * \langle \text{limit multiplier} \rangle$ Note: Check Determine Charging Profile Limit Multiplier for $\langle \text{limit multiplier} \rangle$	
2. The Charging Station responds with a SetChargingProfileResponse	1. The Test System sends a SetChargingProfileRequest with evseld <i><Configured evseld></i> chargingProfile.id <i><Not omitted></i> chargingProfile.stackLevel 0 chargingProfile.chargingProfilePurpose <i>TxDefaultProfile</i> chargingProfile.chargingProfileKind <i>Dynamic</i> chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].startPeriod 0 chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].operationMode <i>ExternalSetpoint</i>
4. The Charging Station responds with a GetChargingProfilesResponse	3. The Test System sends a GetChargingProfilesRequest with evseld <i><Configured evseld></i> chargingProfile.chargingProfilePurpose <i>TxDefaultProfile</i>
5. The Charging Station sends a ReportChargingProfilesRequest	6. The Test System responds with a ReportChargingProfilesResponse
<u>Note(s):</u> - If ReportChargingProfilesRequest.tbc is <i>True</i> in Step 5 then step 5 and 6 will be repeated	
<u>Manual Action:</u> Use external connected system to set setpointReactive to $-8 * \langle \text{limit multiplier} \rangle$ Note: Check Determine Charging Profile Limit Multiplier for $\langle \text{limit multiplier} \rangle$	
8. The Charging Station responds with a GetChargingProfilesResponse	7. The Test System sends a GetChargingProfilesRequest with evseld <i><Configured evseld></i> chargingProfile.chargingProfilePurpose <i>TxDefaultProfile</i>
9. The Charging Station sends a ReportChargingProfilesRequest	10. The Test System responds with a ReportChargingProfilesResponse

Main (Test scenario)Note(s):

- If **ReportChargingProfilesRequest.tbc** is True in Step 9 then step 9 and 10 will be repeated

Tool validations

* Step 2:

Message: **SetChargingProfileResponse**

- **status** must be *Accepted*

* Step 5:

Message **ReportChargingProfilesRequest**

- **requestId** must be *Same Id as in the GetChargingProfilesRequest in step 4*

- **EvseId** must be *<Configured evseId>*

There must be one ChargingProfile be reported with

- **chargingProfile[0].id** must be *<Equal to chargingProfileId step 1>*

- **chargingProfile[0].stackLevel** must be *0*

- **chargingProfile[0].chargingProfilePurpose** must be *TxDefaultProfile*

- **chargingProfile[0].chargingProfileKind** must be *Dynamic*

- **chargingProfile[0].dynUpdateTime** must be *<Datetime of moment of step 7>*

- **chargingProfile[0].chargingSchedule** size must be *1*

- **chargingProfile[0].chargingSchedule[0].chargingSchedulePeriod** size must be *1*

- **chargingProfile[0].chargingSchedule[0].chargingSchedulePeriod[0].startPeriod** must be *0*

- **chargingProfile[0].chargingSchedule[0].chargingSchedulePeriod[0].operationMode** *ExternalSetpoint*

- **chargingProfile[0].chargingSchedule[0].chargingSchedulePeriod[0].setpoint** must be *-6 * <limit multiplier>*

Note: Check [Determine Charging Profile Limit Multiplier](#) for *<limit multiplier>*

* Step 9:

Message **ReportChargingProfilesRequest**

- **requestId** must be *Same Id as in the GetChargingProfilesRequest in step 4*

- **EvseId** must be *<Configured evseId>*

There must be one ChargingProfile be reported with

- **chargingProfile[0].id** must be *<Equal to chargingProfileId step 1>*

- **chargingProfile[0].stackLevel** must be *0*

- **chargingProfile[0].chargingProfilePurpose** must be *TxDefaultProfile*

- **chargingProfile[0].chargingProfileKind** must be *Dynamic*

- **chargingProfile[0].dynUpdateTime** must be *<Datetime of moment of step 7>*

- **chargingProfile[0].chargingSchedule** size must be *1*

- **chargingProfile[0].chargingSchedule[0].chargingSchedulePeriod** size must be *1*

- **chargingProfile[0].chargingSchedule[0].chargingSchedulePeriod[0].startPeriod** must be *0*

- **chargingProfile[0].chargingSchedule[0].chargingSchedulePeriod[0].operationMode** *ExternalSetpoint*

- **chargingProfile[0].chargingSchedule[0].chargingSchedulePeriod[0].setpointReactive** must be *-8 * <limit multiplier>*

Note: Check [Determine Charging Profile Limit Multiplier](#) for *<limit multiplier>*

Post scenario validations:

N/a

TC_Q_112_CS: External V2X control - with a charging profile from CSMS - limit

Test case name	External V2X control - with a charging profile from CSMS - limit
Test case Id	TC_Q_112_CS
Use case Id(s)	Q05
Requirement(s)	Q05.FR.02, Q05.FR.04, Q05.FR.06
System under test	Charging Station
Description	CSMS explicitly allows an External System to control the charge and discharge behaviour of an EV for a certain period of time.
Purpose	To verify if the Charging Station is able to inform the CSMS when it's capacity is being throttled by a locally connected EMS.
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station is connected locally to a system which can apply charging limit constraints. - SmartChargingCtrlr.SupportsFeature[DynamicProfiles] is <i>true</i>

Before (Preparations)
Configuration State: - SmartChargingCtrlr.ExternalControlSignalsEnabled is <i>true</i> - SmartChargingCtrlr.ExternalConstraintsProfileDisallowed is <i>true</i>
Memory State: N/a
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
Manual Action: Use external connected system to set limit 7 and dischargeLimit -6 * <limit multiplier> Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier>	
2. The Charging Station responds with a SetChargingProfileResponse	1. The Test System sends a SetChargingProfileRequest with evseld <Configured evseld> chargingProfile.id <Configured chargingProfileId> chargingProfile.stackLevel 0 chargingProfile.chargingProfilePurpose TxDefaultProfile chargingProfile.chargingProfileKind Dynamic chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].startPeriod 0 chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].operationMode ExternalLimits
4. The Charging Station responds with a GetChargingProfilesResponse	3. The Test System sends a GetChargingProfilesRequest with evseld <Configured evseld> chargingProfile.chargingProfilePurpose TxDefaultProfile
5. The Charging Station sends a ReportChargingProfilesRequest	6. The Test System responds with a ReportChargingProfilesResponse
Note(s): - If ReportChargingProfilesRequest.tbc is True in Step 5 then step 5 and 6 will be repeated	
7. Manual Action: Use external connected system to set limit to 8 * <limit multiplier> and dischargeLimit to 0 Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier>	
9. The Charging Station responds with a GetChargingProfilesResponse	8. The Test System sends a GetChargingProfilesRequest with evseld <Configured evseld> chargingProfile.chargingProfilePurpose TxDefaultProfile
10. The Charging Station sends a ReportChargingProfilesRequest	11. The Test System responds with a ReportChargingProfilesResponse

Main (Test scenario)Note(s):

- If **ReportChargingProfilesRequest.tbc** is True in Step 10 then step 10 and 11 will be repeated

Tool validations

* Step 2:

Message: **SetChargingProfileResponse**

- **status** must be *Accepted*

* Step 5:

Message **ReportChargingProfilesRequest**

- **requestId** must be *Same Id as in the GetChargingProfilesRequest in step 4*

- **EvseId** must be *<Configured evseId>*

There must be one ChargingProfile be reported with

- **chargingProfile[0].id** must be *<Configured chargingProfileId>*

- **chargingProfile[0].stackLevel** must be *0*

- **chargingProfile[0].chargingProfilePurpose** must be *TxDefaultProfile*

- **chargingProfile[0].chargingProfileKind** must be *Dynamic*

- **chargingProfile[0].dynUpdateTime** must be *<Datetime of moment of step 1>*

- **chargingProfile[0].chargingSchedule** size must be *1*

- **chargingProfile[0].chargingSchedule[0].chargingSchedulePeriod** size must be *1*

- **chargingProfile[0].chargingSchedule[0].chargingSchedulePeriod[0].startPeriod** must be *0*

- **chargingProfile[0].chargingSchedule[0].chargingSchedulePeriod[0].operationMode** *ExternalLimits*

- **chargingProfile[0].chargingSchedule[0].chargingSchedulePeriod[0].limit** must be *7 * <limit multiplier>*

- **chargingProfile[0].chargingSchedule[0].chargingSchedulePeriod[0].dischargeLimit** must be *-6 * <limit multiplier>*

Note: Check [Determine Charging Profile Limit Multiplier](#) for *<limit multiplier>*

* Step 10:

Message **ReportChargingProfilesRequest**

- **requestId** must be *Same Id as in the GetChargingProfilesRequest in step 4*

- **EvseId** must be *<Configured evseId>*

There must be one ChargingProfile be reported with

- **chargingProfile[0].id** must be *<Configured chargingProfileId>*

- **chargingProfile[0].stackLevel** must be *0*

- **chargingProfile[0].chargingProfilePurpose** must be *TxDefaultProfile*

- **chargingProfile[0].chargingProfileKind** must be *Dynamic*

- **chargingProfile[0].dynUpdateTime** must be *<Datetime of moment of step 7>*

- **chargingProfile[0].chargingSchedule** size must be *1*

- **chargingProfile[0].chargingSchedule[0].chargingSchedulePeriod** size must be *1*

- **chargingProfile[0].chargingSchedule[0].chargingSchedulePeriod[0].startPeriod** must be *0*

- **chargingProfile[0].chargingSchedule[0].chargingSchedulePeriod[0].operationMode** *ExternalLimits*

- **chargingProfile[0].chargingSchedule[0].chargingSchedulePeriod[0].limit** must be *8 * <limit multiplier>*

- **chargingProfile[0].chargingSchedule[0].chargingSchedulePeriod[0].dischargeLimit** must be *0*

Note: Check [Determine Charging Profile Limit Multiplier](#) for *<limit multiplier>*

Post scenario validations:

N/a

TC_Q_113_CS: External V2X control - with a charging profile from CSMS - Duration expired

Test case name	External V2X control - with a charging profile from CSMS - Duration expired
Test case Id	TC_Q_113_CS
Use case Id(s)	Q05
Requirement(s)	Q05.FR.07
System under test	Charging Station
Description	CSMS explicitly allows an External System to control the charge and discharge behaviour of an EV for a certain period of time.
Purpose	To verify if the Charging Station is able to inform the CSMS when it's capacity is being throttled by a locally connected EMS.
Prerequisite(s)	- The Charging Station is connected locally to a system which can apply charging limit constraints. - SmartChargingCtrlr.SupportsFeature[DynamicProfiles] is <i>true</i>

Before (Preparations)
Configuration State: - SmartChargingCtrlr.ExternalControlSignalsEnabled is <i>true</i> - SmartChargingCtrlr.ExternalConstraintsProfileDisallowed is <i>true</i>
Memory State: N/a
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
Manual Action: Use external connected system to set limit 10 and dischargeLimit 0 * <limit multiplier> Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier>	
2. The Charging Station responds with a SetChargingProfileResponse	1. The Test System sends a SetChargingProfileRequest with evseld <Configured evseld> chargingProfile.id <Configured chargingProfileId> chargingProfile.stackLevel 0 chargingProfile.chargingProfilePurpose TxDefaultProfile chargingProfile.chargingProfileKind Relative chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].startPeriod 0 chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].operationMode ChargingOnly chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].limit 6 * <limit multiplier> Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier>
4. The Charging Station responds with a SetChargingProfileResponse	3. The Test System sends a SetChargingProfileRequest with evseld <Configured evseld> chargingProfile.id <Configured chargingProfileId2> chargingProfile.stackLevel 1 chargingProfile.chargingProfilePurpose TxDefaultProfile chargingProfile.chargingProfileKind Dynamic chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].startPeriod 0 chargingProfile.chargingSchedule[0].duration 30 chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].operationMode ExternalLimits

Main (Test scenario)	
6. The Charging Station responds with a GetCompositeScheduleResponse	5. The Test System sends a GetCompositeScheduleRequest with evseld <Configured evseld> duration 20
<u>Note(s)</u> : The Test System will wait for 35 seconds	
8. The Charging Station responds with a GetCompositeScheduleResponse	7. The Test System sends a GetCompositeScheduleRequest with evseld <Configured evseld> duration 20
Tool validations	
<p>* Step 2: Message: SetChargingProfileResponse - status must be <i>Accepted</i></p> <p>* Step 4: Message: SetChargingProfileResponse - status must be <i>Accepted</i></p> <p>* Step 6: Message: GetCompositeScheduleResponse - status must be <i>Accepted</i> - schedule.chargingSchedulePeriod[0].operationMode is <i>ExternalLimits</i> - schedule.chargingSchedulePeriod[0].limit must be $10 * \text{<limit multiplier>}$ - schedule.chargingSchedulePeriod[0].dischargeLimit must be 0 Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></p> <p>* Step 8: Message: GetCompositeScheduleResponse - status must be <i>Accepted</i> - schedule.chargingSchedulePeriod[0].operationMode is <i>ChargingOnly</i> - schedule.chargingSchedulePeriod[0].limit must be $6 * \text{<limit multiplier>}$ - schedule.chargingSchedulePeriod[0].dischargeLimit must be 0 or absent Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></p>	
Post scenario validations: N/a	

TC_Q_114_CS: External V2X control - with a charging profile from an External System - Dynamic external limits control

Test case name	External V2X control - with a charging profile from an External System - Dynamic external limits control
Test case Id	TC_Q_114_CS
Use case Id(s)	Q06
Requirement(s)	Q06.FR.02, Q06.FR.05, Q06.FR.07, Q06.FR.11
System under test	Charging Station
Description	An External System controls the charge and discharge limits or setpoint of a charging station via a ChargingStationExternalConstraints charging profile.
Purpose	To verify if the Charging Station is able to combine V2X with external control signals.
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station is connected locally to a system which is able to send a single charge limit. - SmartChargingCtrlr.SupportsFeature[DynamicProfiles] is <i>true</i> - EV and charging station support ISO15118-20 or CHAdeMO

Before (Preparations)

Configuration State:

- **SmartChargingCtrlr.LimitChangeSignificance** is 0

Memory State:

N/a

Reusable State(s):

IF <Configured EVProtocol> is CHAdeMO THEN

State is *Authorized*

State is *EVConnectedPreSession*

ELSE IF <Configured EVProtocol> is ISO15118 THEN

State is *Iso1511820V2xControlBpt*

State is *EVConnectedPreSession*

State is *Authorized15118* (for 15118-20)

END IF

Main (Test scenario)

Charging Station	CSMS
1. Manual Action: Use external connected system to send a single limit to the charger: $8 * \text{<limit multiplier>}$ <i>Note:</i> Check <i>Determine Charging Profile Limit Multiplier</i> for <limit multiplier>	
2. The Charging Station sends a NotifyChargingLimitRequest	3. The Test System responds with a NotifyChargingLimitResponse
5. The Charging Station responds with a GetChargingProfilesResponse	4. The Test System sends a GetChargingProfilesRequest with evseld <Configured evseld> chargingProfile.chargingProfilePurpose <i>ChargingStationExternalConstraints</i>
6. The Charging Station sends a ReportChargingProfilesRequest	7. The Test System responds with a ReportChargingProfilesResponse
Note(s): - If ReportChargingProfilesRequest.tbc is True in Step 6 then step 6 and 7 will be repeated	
8. Manual Action: Use external connected system to send a single limit to the charger: $10 * \text{<limit multiplier>}$ <i>Note:</i> Check <i>Determine Charging Profile Limit Multiplier</i> for <limit multiplier>	
9. The Charging Station sends a NotifyChargingLimitRequest	10. The Test System responds with a NotifyChargingLimitResponse

Main (Test scenario)	
12. The Charging Station responds with a GetChargingProfilesResponse	11. The Test System sends a GetChargingProfilesRequest with evseld <Configured evseld> chargingProfile.chargingProfilePurpose <i>ChargingStationExternalConstraints</i>
13. The Charging Station sends a ReportChargingProfilesRequest	14. The Test System responds with a ReportChargingProfilesResponse
<u>Note(s):</u> - If ReportChargingProfilesRequest.tbc is True in Step 13 then step 13 and 14 will be repeated	

Tool validations

* Step 2:

Message: **NotifyChargingLimitRequest**

- **evseld** must be *<Configured evseld>*
- **chargingLimit.chargingLimitSource** must be *EMS*
- **chargingLimit.isLocalGeneration** must be *false*
- **chargingSchedule[0].chargingSchedulePeriod** size must be *1*
- **chargingSchedule[0].chargingSchedulePeriod[0].startPeriod** must be *0*
- **chargingSchedule[0].chargingSchedulePeriod[0].limit** must be $8 * \text{<limit multiplier>}$

Note: Check [Determine Charging Profile Limit Multiplier](#) for *<limit multiplier>*

* Step 5:

Message: **GetChargingProfilesResponse**

- **status** must be *Accepted*

* Step 6:

Message **ReportChargingProfilesRequest**

- **requestId** must be *Same Id as in the GetChargingProfilesRequest in step 4*
- **Evseld** must be *<Configured evseld>*

There must be one ChargingProfile be reported with

- **chargingProfile[0].chargingProfilePurpose** must be *ChargingStationExternalConstraints*
- **chargingProfile[0].chargingProfileKind** must be *Dynamic*
- **chargingProfile[0].dynUpdateTime** must be *<Datetime of moment of step 1>*
- **chargingProfile[0].chargingSchedule** size must be *1*
- **chargingProfile[0].chargingSchedule[0].chargingSchedulePeriod** size must be *1*
- **chargingProfile[0].chargingSchedule[0].chargingSchedulePeriod[0].startPeriod** must be *0*
- **chargingProfile[0].chargingSchedule[0].chargingSchedulePeriod[0].limit** must be $8 * \text{<limit multiplier>}$
- **chargingProfile[0].chargingSchedule[0].chargingSchedulePeriod[0].operationMode** must be *ExternalLimits*

Note: Check [Determine Charging Profile Limit Multiplier](#) for *<limit multiplier>*

* Step 9:

Message: **NotifyChargingLimitRequest**

- **evseld** must be *<Configured evseld>*
- **chargingLimit.chargingLimitSource** must be *EMS*
- **chargingLimit.isLocalGeneration** must be *false*
- **chargingSchedule[0].chargingSchedulePeriod** size must be *1*
- **chargingSchedule[0].chargingSchedulePeriod[0].startPeriod** must be *0*
- **chargingSchedule[0].chargingSchedulePeriod[0].limit** must be $10 * \text{<limit multiplier>}$

Note: Check [Determine Charging Profile Limit Multiplier](#) for *<limit multiplier>*

* Step 12:

Message: **GetChargingProfilesResponse**

- **status** must be *Accepted*

* Step 13:

Message **ReportChargingProfilesRequest**

- **requestId** must be *Same Id as in the GetChargingProfilesRequest in step 11*
- **Evseld** must be *<Configured evseld>*

There must be one ChargingProfile be reported with

- **chargingProfile[0].chargingProfilePurpose** must be *ChargingStationExternalConstraints*
- **chargingProfile[0].chargingProfileKind** must be *Dynamic*
- **chargingProfile[0].dynUpdateTime** must be *<Datetime of moment of step 8>*
- **chargingProfile[0].chargingSchedule** size must be *1*
- **chargingProfile[0].chargingSchedule[0].chargingSchedulePeriod** size must be *1*
- **chargingProfile[0].chargingSchedule[0].chargingSchedulePeriod[0].startPeriod** must be *0*
- **chargingProfile[0].chargingSchedule[0].chargingSchedulePeriod[0].limit** must be $10 * \text{<limit multiplier>}$
- **chargingProfile[0].chargingSchedule[0].chargingSchedulePeriod[0].operationMode** must be *ExternalLimits*

Note: Check [Determine Charging Profile Limit Multiplier](#) for *<limit multiplier>*

Post scenario validations:

N/a

TC_Q_115_CS: External V2X control - with a charging profile from an External System - Dynamic setpoint control

Test case name	External V2X control - with a charging profile from an External System - Dynamic setpoint control
Test case Id	TC_Q_115_CS
Use case Id(s)	Q06
Requirement(s)	Q06.FR.04, Q06.FR.06, Q06.FR.07, Q06.FR.11, Q06.FR.11
System under test	Charging Station
Description	An External System controls the charge and discharge limits or setpoint of a charging station via a ChargingStationExternalConstraints charging profile.
Purpose	To verify if the Charging Station is able to combine V2X with external control signals.
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station is connected locally to a system which is able to send a single charge setpoint. - SmartChargingCtrlr.SupportsFeature[DynamicProfiles] is <i>true</i> - EV and charging station support ISO15118-20 or CHAdeMO

Before (Preparations)
Configuration State: - SmartChargingCtrlr.LimitChangeSignificance is 0
Memory State: N/a
Reusable State(s): IF <Configured EVProtocol> is CHAdeMO THEN State is <i>Authorized</i> State is <i>EVConnectedPreSession</i> ELSE IF <Configured EVProtocol> is ISO15118 THEN State is <i>Iso1511820V2xControlBpt</i> State is <i>EVConnectedPreSession</i> State is <i>Authorized15118</i> (for 15118-20) END IF

Main (Test scenario)	
Charging Station	CSMS
1. <u>Manual Action</u> : Use external connected system to send a single setpoint to the charger: 8 * <limit multiplier> Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier>	
2. The Charging Station sends a NotifyChargingLimitRequest	3. The Test System responds with a NotifyChargingLimitResponse
5. The Charging Station responds with a GetChargingProfilesResponse	4. The Test System sends a GetChargingProfilesRequest with evseld <Configured evseld> chargingProfile.chargingProfilePurpose <i>ChargingStationExternalConstraints</i>
6. The Charging Station sends a ReportChargingProfilesRequest	7. The Test System responds with a ReportChargingProfilesResponse
<u>Note(s)</u> : - If ReportChargingProfilesRequest.tbc is True in Step 6 then step 6 and 7 will be repeated	
8. <u>Manual Action</u> : Use external connected system to send a single setpoint to the charger: 10 * <limit multiplier> Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier>	
9. The Charging Station sends a NotifyChargingLimitRequest	10. The Test System responds with a NotifyChargingLimitResponse

Main (Test scenario)	
12. The Charging Station responds with a GetChargingProfilesResponse	11. The Test System sends a GetChargingProfilesRequest with evseld <Configured evseld> chargingProfile.chargingProfilePurpose <i>ChargingStationExternalConstraints</i>
13. The Charging Station sends a ReportChargingProfilesRequest	14. The Test System responds with a ReportChargingProfilesResponse
<u>Note(s):</u> - If ReportChargingProfilesRequest.tbc is True in Step 13 then step 13 and 14 will be repeated	

Tool validations

* Step 2:

Message: **NotifyChargingLimitRequest**

- **evseld** must be *<Configured evseld>*
- **chargingLimit.chargingLimitSource** must be *EMS*
- **chargingLimit.isLocalGeneration** must be *false*
- **chargingSchedule[0].chargingSchedulePeriod** size must be *1*
- **chargingSchedule[0].chargingSchedulePeriod[0].startPeriod** must be *0*
- **chargingSchedule[0].chargingSchedulePeriod[0].setpoint** must be $8 * \text{<limit multiplier>}$

Note: Check [Determine Charging Profile Limit Multiplier](#) for *<limit multiplier>*

* Step 5:

Message: **GetChargingProfilesResponse**

- **status** must be *Accepted*

* Step 6:

Message **ReportChargingProfilesRequest**

- **requestId** must be *Same Id as in the GetChargingProfilesRequest in step 4*
- **Evseld** must be *<Configured evseld>*

There must be one ChargingProfile be reported with

- **chargingProfile[0].chargingProfilePurpose** must be *ChargingStationExternalConstraints*
- **chargingProfile[0].chargingProfileKind** must be *Dynamic*
- **chargingProfile[0].dynUpdateTime** must be *<Datetime of moment of step 1>*
- **chargingProfile[0].chargingSchedule** size must be *1*
- **chargingProfile[0].chargingSchedule[0].chargingSchedulePeriod** size must be *1*
- **chargingProfile[0].chargingSchedule[0].chargingSchedulePeriod[0].startPeriod** must be *0*
- **chargingProfile[0].chargingSchedule[0].chargingSchedulePeriod[0].setpoint** must be $8 * \text{<limit multiplier>}$
- **chargingProfile[0].chargingSchedule[0].chargingSchedulePeriod[0].operationMode** must be *ExternalLimits*

Note: Check [Determine Charging Profile Limit Multiplier](#) for *<limit multiplier>*

* Step 9:

Message: **NotifyChargingLimitRequest**

- **evseld** must be *<Configured evseld>*
- **chargingLimit.chargingLimitSource** must be *EMS*
- **chargingLimit.isLocalGeneration** must be *false*
- **chargingSchedule[0].chargingSchedulePeriod** size must be *1*
- **chargingSchedule[0].chargingSchedulePeriod[0].startPeriod** must be *0*
- **chargingSchedule[0].chargingSchedulePeriod[0].setpoint** must be $10 * \text{<limit multiplier>}$

Note: Check [Determine Charging Profile Limit Multiplier](#) for *<limit multiplier>*

* Step 12:

Message: **GetChargingProfilesResponse**

- **status** must be *Accepted*

* Step 13:

Message **ReportChargingProfilesRequest**

- **requestId** must be *Same Id as in the GetChargingProfilesRequest in step 11*
- **Evseld** must be *<Configured evseld>*

There must be one ChargingProfile be reported with

- **chargingProfile[0].chargingProfilePurpose** must be *ChargingStationExternalConstraints*
- **chargingProfile[0].chargingProfileKind** must be *Dynamic*
- **chargingProfile[0].dynUpdateTime** must be *<Datetime of moment of step 8>*
- **chargingProfile[0].chargingSchedule** size must be *1*
- **chargingProfile[0].chargingSchedule[0].chargingSchedulePeriod** size must be *1*
- **chargingProfile[0].chargingSchedule[0].chargingSchedulePeriod[0].startPeriod** must be *0*
- **chargingProfile[0].chargingSchedule[0].chargingSchedulePeriod[0].setpoint** must be $10 * \text{<limit multiplier>}$
- **chargingProfile[0].chargingSchedule[0].chargingSchedulePeriod[0].operationMode** must be *ExternalLimits*

Note: Check [Determine Charging Profile Limit Multiplier](#) for *<limit multiplier>*

Post scenario validations:

N/a

TC_Q_116_CS: External V2X control - with a charging profile from an External System - Scheduled external limits control

Test case name	External V2X control - with a charging profile from an External System - Scheduled external limits control
Test case Id	TC_Q_116_CS
Use case Id(s)	Q06
Requirement(s)	Q06.FR.01, Q06.FR.02, Q06.FR.10
System under test	Charging Station
Description	An External System controls the charge and discharge limits or setpoint of a charging station via a ChargingStationExternalConstraints charging profile.
Purpose	To verify if the Charging Station is able to combine V2X with external control signals.
Prerequisite(s)	<ul style="list-style-type: none"> - The Charging Station is connected locally to a system which is able to send a schedule of charge limits. - SmartChargingCtrlr.SupportsFeature[DynamicProfiles] is <i>true</i> - EV and charging station support ISO15118-20 or CHAdeMO

Before (Preparations)
Configuration State: - SmartChargingCtrlr.LimitChangeSignificance is 0
Memory State: N/a
Reusable State(s): IF <Configured EVProtocol> is CHAdeMO THEN State is <i>Authorized</i> State is <i>EVConnectedPreSession</i> ELSE IF <Configured EVProtocol> is ISO15118 THEN State is <i>Iso1511820V2xControlBpt</i> State is <i>EVConnectedPreSession</i> State is <i>Authorized15118</i> (for 15118-20) END IF

Main (Test scenario)	
Charging Station	CSMS
1. <u>Manual Action</u> : Use external connected system to send an Absolute schedule with at least more than one limit to the charger	
2. The Charging Station sends a NotifyChargingLimitRequest	3. The Test System responds with a NotifyChargingLimitResponse
5. The Charging Station responds with a GetChargingProfilesResponse	4. The Test System sends a GetChargingProfilesRequest with evseld <Configured evseld> chargingProfile.chargingProfilePurpose <i>ChargingStationExternalConstraints</i>
6. The Charging Station sends a ReportChargingProfilesRequest	7. The Test System responds with a ReportChargingProfilesResponse
Note(s): - If ReportChargingProfilesRequest.tbc is True in Step 6 then step 6 and 7 will be repeated	
8. <u>Manual Action</u> : Use external connected system to send a Dynamic schedule with at least more than one limit to the charger	
9. The Charging Station sends a NotifyChargingLimitRequest	10. The Test System responds with a NotifyChargingLimitResponse
12. The Charging Station responds with a GetChargingProfilesResponse	11. The Test System sends a GetChargingProfilesRequest with evseld <Configured evseld> chargingProfile.chargingProfilePurpose <i>ChargingStationExternalConstraints</i>

Main (Test scenario)	
13. The Charging Station sends a ReportChargingProfilesRequest	14. The Test System responds with a ReportChargingProfilesResponse
<u>Note(s):</u> - If ReportChargingProfilesRequest.tbc is True in Step 13 then step 13 and 14 will be repeated	

Tool validations
<p>* Step 2:</p> <p>Message: NotifyChargingLimitRequest</p> <ul style="list-style-type: none"> - evseld must be <Configured evseld> - chargingLimit.chargingLimitSource must be <i>EMS</i> - (chargingSchedule.chargingSchedulePeriod[*].limit must be <not omitted> <p>OR chargingSchedule.chargingSchedulePeriod[*].dischargeLimit must be <not omitted>)</p> <p>* Step 5:</p> <p>Message: GetChargingProfilesResponse</p> <ul style="list-style-type: none"> - status must be <i>Accepted</i> <p>* Step 6:</p> <p>Message ReportChargingProfilesRequest</p> <ul style="list-style-type: none"> - requestId must be <i>Same Id as in the GetChargingProfilesRequest in step 4</i> - Evseld must be <Configured evseld> <p>There must be one ChargingProfile be reported with</p> <ul style="list-style-type: none"> - chargingProfile[0].chargingProfilePurpose must be <i>ChargingStationExternalConstraints</i> - chargingProfile[0].chargingProfileKind must be <i>Absolute</i> - chargingProfile[0].chargingSchedule must be <NotifyChargingLimitRequest.chargingSchedule of step 2> - chargingProfile[0].chargingSchedule[0].chargingSchedulePeriod[0].operationMode must be <i>ExternalLimits</i> <p>* Step 9:</p> <p>Message: NotifyChargingLimitRequest</p> <ul style="list-style-type: none"> - evseld must be <Configured evseld> - chargingLimit.chargingLimitSource must be <i>EMS</i> - (chargingSchedule.chargingSchedulePeriod[*].limit must be <not omitted> <p>OR chargingSchedule.chargingSchedulePeriod[*].dischargeLimit must be <not omitted>)</p> <p>* Step 12:</p> <p>Message: GetChargingProfilesResponse</p> <ul style="list-style-type: none"> - status must be <i>Accepted</i> <p>* Step 13:</p> <p>Message ReportChargingProfilesRequest</p> <ul style="list-style-type: none"> - requestId must be <i>Same Id as in the GetChargingProfilesRequest in step 11</i> - Evseld must be <Configured evseld> <p>There must be one ChargingProfile be reported with</p> <ul style="list-style-type: none"> - chargingProfile[0].chargingProfilePurpose must be <i>ChargingStationExternalConstraints</i> - chargingProfile[0].chargingProfileKind must be <i>Dynamic</i> - chargingProfile[0].chargingSchedule must be <NotifyChargingLimitRequest.chargingSchedule of step 9> - chargingProfile[0].chargingSchedule[0].chargingSchedulePeriod[0].operationMode must be <i>ExternalLimits</i> <p>Post scenario validations: N/a</p>

TC_Q_117_CS: Frequency Support - Central V2X control - push

Test case name	Frequency Support - Central V2X control - push
Test case Id	TC_Q_117_CS
Use case Id(s)	Q07
Requirement(s)	K28.FR.05, K28.FR.06, K28.FR.09
System under test	Charging Station
Description	To allow an EV to be used for frequency support, with control at the CSMS.
Purpose	To verify if the Charging Station is able to support DynamicControl charging profiles with the CSMS pushing changes to the CS.
Prerequisite(s)	<ul style="list-style-type: none"> - Charging station supports <i>SupportsDynamicProfiles</i>. - EV and charging station support ISO15118-20 or CHAdeMO - V2XChargingCtrlr.SupportedOperationModes contains <i>LocalFrequency</i>

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): IF <Configured EVProtocol> is CHAdeMO THEN State is <i>Authorized</i> State is <i>EVConnectedPreSession</i> ELSE IF <Configured EVProtocol> is ISO15118 THEN State is <i>Iso1511820V2xControlBpt</i> State is <i>EVConnectedPreSession</i> State is <i>Authorized15118</i> (for 15118-20) END IF State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetChargingProfileResponse	1. The Test System sends a SetChargingProfileRequest with chargingProfile.id 123 chargingProfile.chargingProfilePurpose <i>ChargingStationMaxProfile</i> chargingProfile.chargingProfileKind <i>Dynamic</i> chargingProfile.chargingSchedule[0].chargingRateUnit <Configured <i>chargingRateUnit</i> > chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].startPeriod 0 chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].setpoint 8 * <limit multiplier> Note: Check <i>Determine Charging Profile Limit Multiplier</i> for <limit multiplier> chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].operationMode <i>CentralFrequency</i>
4. The Charging Station responds with a GetChargingProfilesResponse	3. The Test System sends a GetChargingProfilesRequest with chargingProfile.chargingProfileId 123 requestId <Generated <i>requestId</i> >
5. The Charging Station sends a ReportChargingProfilesRequest	6. The Test System responds with a ReportChargingProfilesResponse
Note(s): - If tbc is True at Step 5 then step 5 and 6 will be repeated	

Main (Test scenario)	
8 The Charging Station responds with a UpdateDynamicScheduleResponse	7 The Test System sends a UpdateDynamicScheduleRequest with chargingProfileId 123 scheduleUpdate.setpoint 6 * <limit multiplier> Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier>
10. The Charging Station responds with a GetChargingProfilesResponse	9. The Test System sends a GetChargingProfilesRequest with chargingProfile.chargingProfileId 123 requestId <Generated requestId>
11. The Charging Station sends a ReportChargingProfilesRequest	12. The Test System responds with a ReportChargingProfilesResponse
<u>Note(s):</u> - If tbc is True at Step 11 then step 11 and 12 will be repeated	

Tool validations
<p>* Step 2: Message: SetChargingProfileResponse - status must be Accepted</p> <p>* Step 4: Message: GetChargingProfilesResponse - status must be Accepted</p> <p>* Step 5: Note: The date/time when sending the SetChargingProfileRequest in step 1 is called x: Message: ReportChargingProfilesRequest - chargingProfile.dynUpdateTime must be <x> (+/- Max Time Deviation)</p> <p>* Step 8: Message: UpdateDynamicScheduleResponse - status must be Accepted</p> <p>* Step 10: Message: GetChargingProfilesResponse - status must be Accepted</p> <p>* Step 11: Note: The date/time when sending the UpdateDynamicScheduleRequest in step 7 is called x: Message: ReportChargingProfilesRequest - chargingProfile.dynUpdateTime must be <x> (+/- Max Time Deviation) - chargingProfile.chargingSchedule.chargingSchedulePeriod[0].setpoint must be 6 * <limit multiplier></p> <p>Post scenario validations: N/a</p>

TC_Q_118_CS: Frequency Support - Central V2X control - Duration expired

Test case name	Frequency Support - Central V2X control - Duration expired
Test case Id	TC_Q_118_CS
Use case Id(s)	Q07
Requirement(s)	Q05.FR.07
System under test	Charging Station
Description	To allow an EV to be used for frequency support, with control at the CSMS.
Purpose	To verify if the Charging Station is able to support DynamicControl charging profiles with the CSMS pushing changes to the CS.
Prerequisite(s)	<ul style="list-style-type: none"> - Charging station supports <i>SupportsDynamicProfiles</i>. - EV and charging station support ISO15118-20 or CHAdeMO - V2XChargingCtrlr.SupportedOperationModes contains <i>CentralFrequency</i>

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

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetChargingProfileResponse	1. The Test System sends a SetChargingProfileRequest with evseld <Configured evseld> chargingProfile.id <Configured chargingProfileId> chargingProfile.stackLevel 0 chargingProfile.chargingProfilePurpose TxDefaultProfile chargingProfile.chargingProfileKind Relative chargingProfile.chargingSchedule[0].chargingRateUnit <Configured chargingRateUnit> chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].startPeriod 0 chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].operationMode ChargingOnly chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].limit 6 * <limit multiplier> Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier>

Main (Test scenario)	
4. The Charging Station responds with a SetChargingProfileResponse	<p>3. The Test System sends a SetChargingProfileRequest with</p> <p>evseld <Configured evseld></p> <p>chargingProfile.id <Configured chargingProfileId> + 1</p> <p>chargingProfile.stackLevel 1</p> <p>chargingProfile.chargingProfilePurpose TxDefaultProfile</p> <p>chargingProfile.chargingProfileKind Relative</p> <p>chargingProfile.chargingSchedule[0].chargingRateUnit <Configured chargingRateUnit></p> <p>chargingProfile.chargingSchedule[0].duration 30</p> <p>chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].startPeriod 0</p> <p>chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].setpoint 8 * <limit multiplier></p> <p>Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></p> <p>chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].operationMode CentralFrequency</p>
6. The Charging Station responds with a GetCompositeScheduleResponse	<p>5. The Test System sends a GetCompositeScheduleRequest with</p> <p>evseld <Configured evseld></p> <p>duration 20</p>
<u>Note(s)</u> : The Test System will wait for 35 seconds	
8. The Charging Station responds with a GetCompositeScheduleResponse	<p>7. The Test System sends a GetCompositeScheduleRequest with</p> <p>evseld <Configured evseld></p> <p>duration 20</p>

Tool validations
<p>* Step 2:</p> <p>Message: SetChargingProfileResponse</p> <ul style="list-style-type: none"> - status must be Accepted <p>* Step 4:</p> <p>Message: SetChargingProfileResponse</p> <ul style="list-style-type: none"> - status must be Accepted <p>* Step 6:</p> <p>Message: GetCompositeScheduleResponse</p> <ul style="list-style-type: none"> - status must be Accepted - schedule.chargingSchedulePeriod[0].setpoint must be 8 * <limit multiplier> <p>Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></p> <ul style="list-style-type: none"> - schedule.chargingSchedulePeriod[0].operationMode CentralFrequency <p>* Step 8:</p> <p>Message: GetCompositeScheduleResponse</p> <ul style="list-style-type: none"> - status must be Accepted - schedule.chargingSchedulePeriod[0].limit must be 6 * <limit multiplier> <p>Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></p> <ul style="list-style-type: none"> - schedule.chargingSchedulePeriod[0].operationMode ChargingOnly
<p>Post scenario validations:</p> <p>N/a</p>

TC_Q_119_CS: Frequency Support - Local V2X control - Charging profile validations

Test case name	Frequency Support - Local V2X control - Charging profile validations
Test case Id	TC_Q_119_CS
Use case Id(s)	Q08
Requirement(s)	Q08.FR.01, Q08.FR.03, Q08.FR.04, Q08.FR.05
System under test	Charging Station
Description	To allow an EV to be used for frequency control, depending on local frequency readings.
Purpose	To verify if the Charging Station is able to validate the charging profile for frequency support.
Prerequisite(s)	<ul style="list-style-type: none"> - EV supports V2X power transfer loop - EV and charging station support ISO15118-20 or CHAdeMO - V2XChargingCtrlr.SupportedOperationModes contains <i>LocalFrequency</i>

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): IF <Configured EVProtocol> is CHAdeMO THEN State is Authorized State is EVConnectedPreSession ELSE IF <Configured EVProtocol> is ISO15118 THEN State is Iso1511820V2xControlBpt State is EVConnectedPreSession State is Authorized15118 (for 15118-20) END IF

Main (Test scenario)	
Charging Station	CSMS
Missing v2xBaseline	

Main (Test scenario)	
2. The Charging Station responds with a SetChargingProfileResponse	<p>1. The Test System sends a SetChargingProfileRequest with evseld <i><Configured evseld></i> chargingProfile.transactionId <i><transactionId></i> chargingProfile.chargingProfilePurpose <i>TxProfile</i> chargingProfile.chargingSchedule.chargingRateUnit <i>W</i> chargingProfile.chargingSchedule.chargingSchedulePeriod.operationMode <i>LocalFrequency</i></p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod.limit <i><omitted></i> chargingProfile.chargingSchedule.chargingSchedulePeriod.limit_L2 <i><omitted></i> chargingProfile.chargingSchedule.chargingSchedulePeriod.limit_L3 <i><omitted></i></p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod.dischargeLimit <i><omitted></i> chargingProfile.chargingSchedule.chargingSchedulePeriod.dischargeLimit_L2 <i><omitted></i> chargingProfile.chargingSchedule.chargingSchedulePeriod.dischargeLimit_L3 <i><omitted></i></p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod.setpoint <i><omitted></i> chargingProfile.chargingSchedule.chargingSchedulePeriod.setpoint_L2 <i><omitted></i> chargingProfile.chargingSchedule.chargingSchedulePeriod.setpoint_L3 <i><omitted></i></p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod.setpointReactive <i><omitted></i> chargingProfile.chargingSchedule.chargingSchedulePeriod.setpointReactive_L2 <i><omitted></i> chargingProfile.chargingSchedule.chargingSchedulePeriod.setpointReactive_L3 <i><omitted></i></p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod.v2xFreqWattCurve[0].frequency <i><not omitted></i> chargingProfile.chargingSchedule.chargingSchedulePeriod.v2xFreqWattCurve[0].power <i><not omitted></i> chargingProfile.chargingSchedule.chargingSchedulePeriod.v2xFreqWattCurve[1].frequency <i><not omitted></i> chargingProfile.chargingSchedule.chargingSchedulePeriod.v2xFreqWattCurve[1].power <i><not omitted></i> chargingProfile.chargingSchedule.chargingSchedulePeriod.v2xBaseline <i><omitted></i></p>
Missing both v2xFreqWattCurves	

Main (Test scenario)	
<p>4. The Charging Station responds with a SetChargingProfileResponse</p>	<p>3. The Test System sends a SetChargingProfileRequest with evseld <Configured evseld> chargingProfile.transactionId <transactionId> chargingProfile.chargingProfilePurpose TxProfile chargingProfile.chargingSchedule.chargingRateUnit W chargingProfile.chargingSchedule.chargingSchedulePeriod.operationMode LocalFrequency</p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod.limit <omitted> chargingProfile.chargingSchedule.chargingSchedulePeriod.limit_L2 <omitted> chargingProfile.chargingSchedule.chargingSchedulePeriod.limit_L3 <omitted></p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod.dischargeLimit <omitted> chargingProfile.chargingSchedule.chargingSchedulePeriod.dischargeLimit_L2 <omitted> chargingProfile.chargingSchedule.chargingSchedulePeriod.dischargeLimit_L3 <omitted></p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod.setpoint <omitted> chargingProfile.chargingSchedule.chargingSchedulePeriod.setpoint_L2 <omitted> chargingProfile.chargingSchedule.chargingSchedulePeriod.setpoint_L3 <omitted></p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod.setpointReactive <omitted> chargingProfile.chargingSchedule.chargingSchedulePeriod.setpointReactive_L2 <omitted> chargingProfile.chargingSchedule.chargingSchedulePeriod.setpointReactive_L3 <omitted></p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod.v2xFreqWattCurve[0].frequency <omitted> chargingProfile.chargingSchedule.chargingSchedulePeriod.v2xFreqWattCurve[0].power <omitted> chargingProfile.chargingSchedule.chargingSchedulePeriod.v2xFreqWattCurve[1].frequency <omitted> chargingProfile.chargingSchedule.chargingSchedulePeriod.v2xFreqWattCurve[1].power <omitted> chargingProfile.chargingSchedule.chargingSchedulePeriod.v2xBaseline <not omitted></p>
Missing one v2xFreqWattCurve	

Main (Test scenario)	
<p>6. The Charging Station responds with a SetChargingProfileResponse</p>	<p>5. The Test System sends a SetChargingProfileRequest with evseld <Configured evseld> chargingProfile.transactionId <transactionId> chargingProfile.chargingProfilePurpose TxProfile chargingProfile.chargingSchedule.chargingRateUnit W chargingProfile.chargingSchedule.chargingSchedulePeriod.operationMode LocalFrequency</p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod.limit <omitted> chargingProfile.chargingSchedule.chargingSchedulePeriod.limit_L2 <omitted> chargingProfile.chargingSchedule.chargingSchedulePeriod.limit_L3 <omitted></p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod.dischargeLimit <omitted> chargingProfile.chargingSchedule.chargingSchedulePeriod.dischargeLimit_L2 <omitted> chargingProfile.chargingSchedule.chargingSchedulePeriod.dischargeLimit_L3 <omitted></p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod.setpoint <omitted> chargingProfile.chargingSchedule.chargingSchedulePeriod.setpoint_L2 <omitted> chargingProfile.chargingSchedule.chargingSchedulePeriod.setpoint_L3 <omitted></p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod.setpointReactive <omitted> chargingProfile.chargingSchedule.chargingSchedulePeriod.setpointReactive_L2 <omitted> chargingProfile.chargingSchedule.chargingSchedulePeriod.setpointReactive_L3 <omitted></p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod.v2xFreqWattCurve[0].frequency <not omitted> chargingProfile.chargingSchedule.chargingSchedulePeriod.v2xFreqWattCurve[0].power <not omitted> chargingProfile.chargingSchedule.chargingSchedulePeriod.v2xFreqWattCurve[1].frequency <omitted> chargingProfile.chargingSchedule.chargingSchedulePeriod.v2xFreqWattCurve[1].power <omitted> chargingProfile.chargingSchedule.chargingSchedulePeriod.v2xBaseline <not omitted></p>
Contains limit fields	

Main (Test scenario)	
<p>8. The Charging Station responds with a SetChargingProfileResponse</p>	<p>7. The Test System sends a SetChargingProfileRequest with evseld <Configured evseld> chargingProfile.transactionId <transactionId> chargingProfile.chargingProfilePurpose TxProfile chargingProfile.chargingSchedule.chargingRateUnit W chargingProfile.chargingSchedule.chargingSchedulePeriod.operationMode LocalFrequency</p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod.limit <not omitted> chargingProfile.chargingSchedule.chargingSchedulePeriod.limit_L2 <not omitted> chargingProfile.chargingSchedule.chargingSchedulePeriod.limit_L3 <not omitted></p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod.dischargeLimit <omitted> chargingProfile.chargingSchedule.chargingSchedulePeriod.dischargeLimit_L2 <omitted> chargingProfile.chargingSchedule.chargingSchedulePeriod.dischargeLimit_L3 <omitted></p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod.setpoint <omitted> chargingProfile.chargingSchedule.chargingSchedulePeriod.setpoint_L2 <omitted> chargingProfile.chargingSchedule.chargingSchedulePeriod.setpoint_L3 <omitted></p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod.setpointReactive <omitted> chargingProfile.chargingSchedule.chargingSchedulePeriod.setpointReactive_L2 <omitted> chargingProfile.chargingSchedule.chargingSchedulePeriod.setpointReactive_L3 <omitted></p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod.v2xFreqWattCurve[0].frequency <not omitted> chargingProfile.chargingSchedule.chargingSchedulePeriod.v2xFreqWattCurve[0].power <not omitted> chargingProfile.chargingSchedule.chargingSchedulePeriod.v2xFreqWattCurve[1].frequency <not omitted> chargingProfile.chargingSchedule.chargingSchedulePeriod.v2xFreqWattCurve[1].power <not omitted> chargingProfile.chargingSchedule.chargingSchedulePeriod.v2xBaseline <not omitted></p>
Contains <i>dischargingLimit</i> fields	

Main (Test scenario)	
<p>10. The Charging Station responds with a SetChargingProfileResponse</p>	<p>9. The Test System sends a SetChargingProfileRequest with evseld <Configured evseld> chargingProfile.transactionId <transactionId> chargingProfile.chargingProfilePurpose TxProfile chargingProfile.chargingSchedule.chargingRateUnit W chargingProfile.chargingSchedule.chargingSchedulePeriod.operationMode LocalFrequency</p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod.limit <omitted> chargingProfile.chargingSchedule.chargingSchedulePeriod.limit_L2 <omitted> chargingProfile.chargingSchedule.chargingSchedulePeriod.limit_L3 <omitted></p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod.dischargeLimit <not omitted> chargingProfile.chargingSchedule.chargingSchedulePeriod.dischargeLimit_L2 <not omitted> chargingProfile.chargingSchedule.chargingSchedulePeriod.dischargeLimit_L3 <not omitted></p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod.setpoint <omitted> chargingProfile.chargingSchedule.chargingSchedulePeriod.setpoint_L2 <omitted> chargingProfile.chargingSchedule.chargingSchedulePeriod.setpoint_L3 <omitted></p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod.setpointReactive <omitted> chargingProfile.chargingSchedule.chargingSchedulePeriod.setpointReactive_L2 <omitted> chargingProfile.chargingSchedule.chargingSchedulePeriod.setpointReactive_L3 <omitted></p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod.v2xFreqWattCurve[0].frequency <not omitted> chargingProfile.chargingSchedule.chargingSchedulePeriod.v2xFreqWattCurve[0].power <not omitted> chargingProfile.chargingSchedule.chargingSchedulePeriod.v2xFreqWattCurve[1].frequency <not omitted> chargingProfile.chargingSchedule.chargingSchedulePeriod.v2xFreqWattCurve[1].power <not omitted> chargingProfile.chargingSchedule.chargingSchedulePeriod.v2xBaseline <not omitted></p>
Contains setpoint fields	

Main (Test scenario)	
12. The Charging Station responds with a SetChargingProfileResponse	<p>11. The Test System sends a SetChargingProfileRequest with evseld <Configured evseld> chargingProfile.transactionId <transactionId> chargingProfile.chargingProfilePurpose TxProfile chargingProfile.chargingSchedule.chargingRateUnit W chargingProfile.chargingSchedule.chargingSchedulePeriod.operationMode LocalFrequency</p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod.limit <omitted> chargingProfile.chargingSchedule.chargingSchedulePeriod.limit_L2 <omitted> chargingProfile.chargingSchedule.chargingSchedulePeriod.limit_L3 <omitted></p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod.dischargeLimit <omitted> chargingProfile.chargingSchedule.chargingSchedulePeriod.dischargeLimit_L2 <omitted> chargingProfile.chargingSchedule.chargingSchedulePeriod.dischargeLimit_L3 <omitted></p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod.setpoint <not omitted> chargingProfile.chargingSchedule.chargingSchedulePeriod.setpoint_L2 <not omitted> chargingProfile.chargingSchedule.chargingSchedulePeriod.setpoint_L3 <not omitted></p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod.setpointReactive <omitted> chargingProfile.chargingSchedule.chargingSchedulePeriod.setpointReactive_L2 <omitted> chargingProfile.chargingSchedule.chargingSchedulePeriod.setpointReactive_L3 <omitted></p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod.v2xFreqWattCurve[0].frequency <not omitted> chargingProfile.chargingSchedule.chargingSchedulePeriod.v2xFreqWattCurve[0].power <not omitted> chargingProfile.chargingSchedule.chargingSchedulePeriod.v2xFreqWattCurve[1].frequency <not omitted> chargingProfile.chargingSchedule.chargingSchedulePeriod.v2xFreqWattCurve[1].power <not omitted> chargingProfile.chargingSchedule.chargingSchedulePeriod.v2xBaseline <not omitted></p>
Contains setpointReactive fields	

Main (Test scenario)	
<p>14. The Charging Station responds with a SetChargingProfileResponse status <i>Rejected</i> statusInfo.reasonCode <i>InvalidSchedule</i></p>	<p>13. The Test System sends a SetChargingProfileRequest with evseld <i><Configured evseld></i> chargingProfile.transactionId <i><transactionId></i> chargingProfile.chargingProfilePurpose <i>TxProfile</i> chargingProfile.chargingSchedule.chargingRateUnit <i>W</i> chargingProfile.chargingSchedule.chargingSchedulePeriod.operationMode <i>LocalFrequency</i></p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod.limit <i><omitted></i> chargingProfile.chargingSchedule.chargingSchedulePeriod.limit_L2 <i><omitted></i> chargingProfile.chargingSchedule.chargingSchedulePeriod.limit_L3 <i><omitted></i></p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod.dischargeLimit <i><omitted></i> chargingProfile.chargingSchedule.chargingSchedulePeriod.dischargeLimit_L2 <i><omitted></i> chargingProfile.chargingSchedule.chargingSchedulePeriod.dischargeLimit_L3 <i><omitted></i></p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod.setpoint <i><omitted></i> chargingProfile.chargingSchedule.chargingSchedulePeriod.setpoint_L2 <i><omitted></i> chargingProfile.chargingSchedule.chargingSchedulePeriod.setpoint_L3 <i><omitted></i></p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod.setpointReactive <i><not omitted></i> chargingProfile.chargingSchedule.chargingSchedulePeriod.setpointReactive_L2 <i><not omitted></i> chargingProfile.chargingSchedule.chargingSchedulePeriod.setpointReactive_L3 <i><not omitted></i></p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod.v2xFreqWattCurve[0].frequency <i><not omitted></i> chargingProfile.chargingSchedule.chargingSchedulePeriod.v2xFreqWattCurve[0].power <i><not omitted></i> chargingProfile.chargingSchedule.chargingSchedulePeriod.v2xFreqWattCurve[1].frequency <i><not omitted></i> chargingProfile.chargingSchedule.chargingSchedulePeriod.v2xFreqWattCurve[1].power <i><not omitted></i> chargingProfile.chargingSchedule.chargingSchedulePeriod.v2xBaseline <i><not omitted></i></p>
All ok	

Main (Test scenario)	
<p>16. The Charging Station responds with a SetChargingProfileResponse with status must be <i>Accepted</i></p>	<p>15. The Test System sends a SetChargingProfileRequest with evseld <i><Configured evseld></i> chargingProfile.transactionId <i><transactionId></i> chargingProfile.chargingProfilePurpose <i>TxProfile</i> chargingProfile.chargingSchedule.chargingRateUnit <i>W</i> chargingProfile.chargingSchedule.chargingSchedulePeriod.operationMode <i>LocalFrequency</i></p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod.limit <i><omitted></i> chargingProfile.chargingSchedule.chargingSchedulePeriod.limit_L2 <i><omitted></i> chargingProfile.chargingSchedule.chargingSchedulePeriod.limit_L3 <i><omitted></i></p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod.dischargeLimit <i><omitted></i> chargingProfile.chargingSchedule.chargingSchedulePeriod.dischargeLimit_L2 <i><omitted></i> chargingProfile.chargingSchedule.chargingSchedulePeriod.dischargeLimit_L3 <i><omitted></i></p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod.setpoint <i><omitted></i> chargingProfile.chargingSchedule.chargingSchedulePeriod.setpoint_L2 <i><omitted></i> chargingProfile.chargingSchedule.chargingSchedulePeriod.setpoint_L3 <i><omitted></i></p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod.setpointReactive <i><omitted></i> chargingProfile.chargingSchedule.chargingSchedulePeriod.setpointReactive_L2 <i><omitted></i> chargingProfile.chargingSchedule.chargingSchedulePeriod.setpointReactive_L3 <i><omitted></i></p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod.v2xFreqWattCurve[0].frequency <i><not omitted></i> chargingProfile.chargingSchedule.chargingSchedulePeriod.v2xFreqWattCurve[0].power <i><not omitted></i> chargingProfile.chargingSchedule.chargingSchedulePeriod.v2xFreqWattCurve[1].frequency <i><not omitted></i> chargingProfile.chargingSchedule.chargingSchedulePeriod.v2xFreqWattCurve[1].power <i><not omitted></i> chargingProfile.chargingSchedule.chargingSchedulePeriod.v2xBaseline <i><not omitted></i></p>

Tool validations
<p>* Step 2:</p> <p>Message: SetChargingProfileResponse</p> <ul style="list-style-type: none"> - status must be <i>Rejected</i> - statusInfo.reasonCode must be <i>NoFreqWattCurve</i> <p>* Step 4:</p> <p>Message: SetChargingProfileResponse</p> <ul style="list-style-type: none"> - status must be <i>Rejected</i> - statusInfo.reasonCode must be <i>NoFreqWattCurve</i> <p>* Step 6:</p> <p>Message: SetChargingProfileResponse</p> <ul style="list-style-type: none"> - status must be <i>Rejected</i> - statusInfo.reasonCode must be <i>NoFreqWattCurve</i> <p>* Step 8:</p> <p>Message: SetChargingProfileResponse</p> <ul style="list-style-type: none"> - status must be <i>Rejected</i> - statusInfo.reasonCode must be <i>InvalidSchedule</i> <p>* Step 10:</p> <p>Message: SetChargingProfileResponse</p> <ul style="list-style-type: none"> - status must be <i>Rejected</i> - statusInfo.reasonCode must be <i>InvalidSchedule</i> <p>* Step 12:</p> <p>Message: SetChargingProfileResponse</p> <ul style="list-style-type: none"> - status must be <i>Rejected</i> - statusInfo.reasonCode must be <i>InvalidSchedule</i> <p>* Step 14:</p> <p>Message: SetChargingProfileResponse</p> <ul style="list-style-type: none"> - status must be <i>Rejected</i> - statusInfo.reasonCode must be <i>InvalidSchedule</i> <p>* Step 16:</p> <p>Message: SetChargingProfileResponse</p> <ul style="list-style-type: none"> - status must be <i>Accepted</i>
<p>Post scenario validations:</p> <p>N/a</p>

TC_Q_120_CS: Frequency Support - Local V2X control - AFRR support

Test case name	Frequency Support - Local V2X control - AFRR support
Test case Id	TC_Q_120_CS
Use case Id(s)	Q08
Requirement(s)	Q08.FR.11
System under test	Charging Station
Description	To allow an EV to be used for frequency control, depending on local frequency readings.
Purpose	To verify if the Charging Station is able to validate the charging profile for frequency support.
Prerequisite(s)	<ul style="list-style-type: none"> - EV supports V2X power transfer loop - EV and charging station support ISO15118-20 or CHAdeMO - V2XChargingCtrlr.SupportedOperationModes contains <i>LocalFrequency</i>

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): IF <Configured EVProtocol> is <i>CHAdeMO</i> THEN State is <i>Authorized</i> State is <i>EVConnectedPreSession</i> ELSE IF <Configured EVProtocol> is <i>ISO15118</i> THEN State is <i>Iso1511820V2xControlBpt</i> State is <i>EVConnectedPreSession</i> State is <i>Authorized15118</i> (for 15118-20) END IF

Main (Test scenario)	
Charging Station	CSMS
No LocalFrequency profile	
2. The Charging Station responds with a SetChargingProfileResponse	1. The Test System sends a SetChargingProfileRequest with evseld <Configured evseld> chargingProfile <ul style="list-style-type: none">- transactionId <transactionId>- chargingProfilePurpose TxProfile- chargingSchedule.chargingRateUnit W- chargingSchedule.chargingSchedulePeriod.operationMode ChargingOnly
4. The Charging Station responds with AFRRSignalResponse	3. The Test System sends a AFRRSignalRequest
No v2xSignalWattCurve set in profile	

Main (Test scenario)	
6. The Charging Station responds with a SetChargingProfileResponse	<p>5. The Test System sends a SetChargingProfileRequest with evseld <Configured evseld></p> <p>chargingProfile</p> <ul style="list-style-type: none"> - transactionId <transactionId> - chargingProfilePurpose TxProfile - chargingSchedule.chargingRateUnit W - chargingSchedule.chargingSchedulePeriod.operationMode LocalFrequency - chargingSchedule.chargingSchedulePeriod.limit <omitted> - chargingSchedule.chargingSchedulePeriod.limit_L2 <omitted> - chargingSchedule.chargingSchedulePeriod.limit_L3 <omitted> - chargingSchedule.chargingSchedulePeriod.dischargeLimit <omitted> - chargingSchedule.chargingSchedulePeriod.dischargeLimit_L2 <omitted> - chargingSchedule.chargingSchedulePeriod.dischargeLimit_L3 <omitted> - chargingSchedule.chargingSchedulePeriod.setpoint <omitted> - chargingSchedule.chargingSchedulePeriod.setpoint_L2 <omitted> - chargingSchedule.chargingSchedulePeriod.setpoint_L3 <omitted> - chargingSchedule.chargingSchedulePeriod.setpointReactive <omitted> - chargingSchedule.chargingSchedulePeriod.setpointReactive_L2 <omitted> - chargingSchedule.chargingSchedulePeriod.setpointReactive_L3 <omitted> - chargingSchedule.chargingSchedulePeriod.v2xFreqWattCurve[0].frequency <not omitted> - chargingSchedule.chargingSchedulePeriod.v2xFreqWattCurve[0].power <not omitted> - chargingSchedule.chargingSchedulePeriod.v2xFreqWattCurve[1].frequency <not omitted> - chargingSchedule.chargingSchedulePeriod.v2xFreqWattCurve[1].power <not omitted> - chargingSchedule.chargingSchedulePeriod.v2xBaseline <not omitted> - chargingSchedule.chargingSchedulePeriod.v2xSignalWattCurve <omitted>
8. The Charging Station responds with AFRRSignalResponse	7. The Test System sends a AFRRSignalRequest
All ok	

Main (Test scenario)	
<p>10. The Charging Station responds with a SetChargingProfileResponse</p>	<p>9. The Test System sends a SetChargingProfileRequest with evseld <Configured evseld></p> <p>chargingProfile</p> <ul style="list-style-type: none"> - transactionId <transactionId> - chargingProfilePurpose TxProfile - chargingSchedule.chargingRateUnit W - chargingSchedule.chargingSchedulePeriod.operationMode LocalFrequency - chargingSchedule.chargingSchedulePeriod.limit <omitted> - chargingSchedule.chargingSchedulePeriod.limit_L2 <omitted> - chargingSchedule.chargingSchedulePeriod.limit_L3 <omitted> - chargingSchedule.chargingSchedulePeriod.dischargeLimit <omitted> - chargingSchedule.chargingSchedulePeriod.dischargeLimit_L2 <omitted> - chargingSchedule.chargingSchedulePeriod.dischargeLimit_L3 <omitted> - chargingSchedule.chargingSchedulePeriod.setpoint <omitted> - chargingSchedule.chargingSchedulePeriod.setpoint_L2 <omitted> - chargingSchedule.chargingSchedulePeriod.setpoint_L3 <omitted> - chargingSchedule.chargingSchedulePeriod.setpointReactive <omitted> - chargingSchedule.chargingSchedulePeriod.setpointReactive_L2 <omitted> - chargingSchedule.chargingSchedulePeriod.setpointReactive_L3 <omitted> - chargingSchedule.chargingSchedulePeriod.v2xFreqWattCurve[0].frequency <not omitted> - chargingSchedule.chargingSchedulePeriod.v2xFreqWattCurve[0].power <not omitted> - chargingSchedule.chargingSchedulePeriod.v2xFreqWattCurve[1].frequency <not omitted> - chargingSchedule.chargingSchedulePeriod.v2xFreqWattCurve[1].power <not omitted> - chargingSchedule.chargingSchedulePeriod.v2xBaseline <not omitted> - chargingSchedule.chargingSchedulePeriod.v2xSignalWattCurve <not omitted>
<p>12. The Charging Station responds with AFRRSignalResponse</p>	<p>11. The Test System sends a AFRRSignalRequest</p>

Tool validations
<div><div>* Step 2:</div><div>Message: SetChargingProfileResponse</div><div>- status must be <i>Accepted</i></div><div>* Step 4:</div><div>Message: AFRRSignalResponse</div><div>- status must be <i>Rejected</i></div><div>- statusInfo.reasonCode must be <i>NoSignalWattCurve</i></div><div>* Step 6:</div><div>Message: SetChargingProfileResponse</div><div>- status must be <i>Accepted</i></div><div>* Step 8:</div><div>Message: AFRRSignalResponse</div><div>- status must be <i>Rejected</i></div><div>- statusInfo.reasonCode must be <i>NoSignalWattCurve</i></div><div>* Step 10:</div><div>Message: SetChargingProfileResponse</div><div>- status must be <i>Accepted</i></div><div>* Step 12:</div><div>Message: AFRRSignalResponse</div><div>- status must be <i>Accepted</i></div></div>
<div>Post scenario validations:</div> <div>N/a</div>

TC_Q_122_CS: Local V2X control for load balancing - threshold validations

Test case name	Local V2X control for load balancing - threshold validations
Test case Id	TC_Q_122_CS
Use case Id(s)	Q09
Requirement(s)	Q09.FR.02
System under test	Charging Station
Description	To allow the EV to be utilized for locally controlled load balancing.
Purpose	To verify that the Charging Station for which LocalLoadBalancing has not correctly configured, rejects charging profiles with operationMode LocalLoadBalancing.
Prerequisite(s)	<ul style="list-style-type: none"> - EV supports V2X power transfer loop - EV and charging station support ISO15118-20 or CHAdeMO - V2XChargingCtrlr.SupportedOperationModes contains <i>LocalLoadBalancing</i>

Before (Preparations)
<p>Configuration State:</p> <ul style="list-style-type: none"> - V2XChargingCtrlr.V2XLocalLoadBalancing[UpperThreshold] is 2000 - V2XChargingCtrlr.V2XLocalLoadBalancing[LowerThreshold] is 1000 - V2XChargingCtrlr.V2XLocalLoadBalancing[UpperOffset] is 200 - V2XChargingCtrlr.V2XLocalLoadBalancing[LowerOffset] is 300 <p>Memory State: N/a</p> <p>Reusable State(s): IF <Configured EVProtocol> is CHAdeMO THEN State is <i>Authorized</i> State is <i>EVConnectedPreSession</i> ELSE IF <Configured EVProtocol> is ISO15118 THEN State is <i>Iso1511820V2xControlBpt</i> State is <i>EVConnectedPreSession</i> State is <i>Authorized15118</i> (for 15118-20) END IF</p>

Main (Test scenario)	
Charging Station	CSMS
Missing UpperThreshold	
2. The Charging Station responds with a SetVariablesResponse	1. The Test System sends a SetVariablesRequest with setVariableData[0].component.name <i>V2XChargingCtrlr</i> setVariableData[0].variable.name <i>V2XLocalLoadBalancing</i> setVariableData[0].variable.instance <i>UpperThreshold</i> setVariableData[0].attributeValue ""
4. The Charging Station responds with a SetChargingProfileResponse	3. The Test System sends a SetChargingProfileRequest with evseld <i><Configured evseld></i> chargingProfile - transactionId <i><transactionId></i> - chargingProfilePurpose <i>TxProfile</i> - chargingSchedule.chargingRateUnit <i>W</i> - chargingSchedule.chargingSchedulePeriod.operationMode <i>LocalLoadBalancing</i>
Missing LowerThreshold	

Main (Test scenario)	
6. The Charging Station responds with a SetVariablesResponse	5. The Test System sends a SetVariablesRequest with setVariableData[0].component.name <i>V2XChargingCtrlr</i> setVariableData[0].variable.name <i>V2XLocalLoadBalancing</i> setVariableData[0].variable.instance <i>UpperThreshold</i> setVariableData[0].attributeValue <i>2000</i> setVariableData[1].component.name <i>V2XChargingCtrlr</i> setVariableData[1].variable.name <i>V2XLocalLoadBalancing</i> setVariableData[1].variable.instance <i>LowerThreshold</i> setVariableData[1].attributeValue <i>""</i>
8. The Charging Station responds with a SetChargingProfileResponse	7. The Test System sends a SetChargingProfileRequest with evseld <i><Configured evseld></i> chargingProfile - transactionId <i><transactionId></i> - chargingProfilePurpose <i>TxProfile</i> - chargingSchedule.chargingRateUnit <i>W</i> - chargingSchedule.chargingSchedulePeriod.operationMode <i>LocalLoadBalancing</i>
Missing UpperOffset	
10. The Charging Station responds with a SetVariablesResponse	9. The Test System sends a SetVariablesRequest with setVariableData[0].component.name <i>V2XChargingCtrlr</i> setVariableData[0].variable.name <i>V2XLocalLoadBalancing</i> setVariableData[0].variable.instance <i>LowerThreshold</i> setVariableData[0].attributeValue <i>1000</i> setVariableData[1].component.name <i>V2XChargingCtrlr</i> setVariableData[1].variable.name <i>V2XLocalLoadBalancing</i> setVariableData[1].variable.instance <i>UpperOffset</i> setVariableData[1].attributeValue <i>""</i>
12. The Charging Station responds with a SetChargingProfileResponse	11. The Test System sends a SetChargingProfileRequest with evseld <i><Configured evseld></i> chargingProfile - transactionId <i><transactionId></i> - chargingProfilePurpose <i>TxProfile</i> - chargingSchedule.chargingRateUnit <i>W</i> - chargingSchedule.chargingSchedulePeriod.operationMode <i>LocalLoadBalancing</i>
Missing LowerOffset	
14. The Charging Station responds with a SetVariablesResponse	13. The Test System sends a SetVariablesRequest with setVariableData[0].component.name <i>V2XChargingCtrlr</i> setVariableData[0].variable.name <i>V2XLocalLoadBalancing</i> setVariableData[0].variable.instance <i>UpperOffset</i> setVariableData[0].attributeValue <i>200</i> setVariableData[1].component.name <i>V2XChargingCtrlr</i> setVariableData[1].variable.name <i>V2XLocalLoadBalancing</i> setVariableData[1].variable.instance <i>LowerOffset</i> setVariableData[1].attributeValue <i>""</i>
16. The Charging Station responds with a SetChargingProfileResponse	15. The Test System sends a SetChargingProfileRequest with evseld <i><Configured evseld></i> chargingProfile - transactionId <i><transactionId></i> - chargingProfilePurpose <i>TxProfile</i> - chargingSchedule.chargingRateUnit <i>W</i> - chargingSchedule.chargingSchedulePeriod.operationMode <i>LocalLoadBalancing</i>

Main (Test scenario)	
<i>UpperThreshold < LowerThreshold</i>	
18. The Charging Station responds with a SetVariablesResponse	17. The Test System sends a SetVariablesRequest with setVariableData[0].component.name <i>V2XChargingCtrlr</i> setVariableData[0].variable.name <i>V2XLocalLoadBalancing</i> setVariableData[0].variable.instance <i>LowerOffset</i> setVariableData[0].attributeValue <i>300</i> setVariableData[1].component.name <i>V2XChargingCtrlr</i> setVariableData[1].variable.name <i>V2XLocalLoadBalancing</i> setVariableData[1].variable.instance <i>LowerThreshold</i> setVariableData[1].attributeValue <i>3000</i>
20. The Charging Station responds with a SetChargingProfileResponse	19. The Test System sends a SetChargingProfileRequest with evseld <i><Configured evseld></i> chargingProfile - transactionId <i><transactionId></i> - chargingProfilePurpose <i>TxProfile</i> - chargingSchedule.chargingRateUnit <i>W</i> - chargingSchedule.chargingSchedulePeriod.operationMode <i>LocalLoadBalancing</i>
<i>All ok</i>	
22. The Charging Station responds with a SetVariablesResponse	21. The Test System sends a SetVariablesRequest with setVariableData[0].component.name <i>V2XChargingCtrlr</i> setVariableData[0].variable.name <i>V2XLocalLoadBalancing</i> setVariableData[0].variable.instance <i>LowerThreshold</i> setVariableData[0].attributeValue <i>1000</i>
24. The Charging Station responds with a SetChargingProfileResponse	23. The Test System sends a SetChargingProfileRequest with evseld <i><Configured evseld></i> chargingProfile - transactionId <i><transactionId></i> - chargingProfilePurpose <i>TxProfile</i> - chargingSchedule.chargingRateUnit <i>W</i> - chargingSchedule.chargingSchedulePeriod.operationMode <i>LocalLoadBalancing</i>

Tool validations
<p>* Step 2:</p> <p>Message: SetVariablesResponse</p> <ul style="list-style-type: none"> - setVariableResult[0].attributeStatus must be <i>Accepted</i> <p>* Step 4:</p> <p>Message: SetChargingProfileResponse</p> <ul style="list-style-type: none"> - status must be <i>Rejected</i> - statusInfo.reasonCode must be <i>MissingDevModelInfo</i> <p>* Step 6:</p> <p>Message: SetVariablesResponse</p> <ul style="list-style-type: none"> - setVariableResult[0].attributeStatus must be <i>Accepted</i> - setVariableResult[1].attributeStatus must be <i>Accepted</i> <p>* Step 8:</p> <p>Message: SetChargingProfileResponse</p> <ul style="list-style-type: none"> - status must be <i>Rejected</i> - statusInfo.reasonCode must be <i>MissingDevModelInfo</i> <p>* Step 10:</p> <p>Message: SetVariablesResponse</p> <ul style="list-style-type: none"> - setVariableResult[0].attributeStatus must be <i>Accepted</i> - setVariableResult[1].attributeStatus must be <i>Accepted</i> <p>* Step 12:</p> <p>Message: SetChargingProfileResponse</p> <ul style="list-style-type: none"> - status must be <i>Rejected</i> - statusInfo.reasonCode must be <i>MissingDevModelInfo</i> <p>* Step 14:</p> <p>Message: SetVariablesResponse</p> <ul style="list-style-type: none"> - setVariableResult[0].attributeStatus must be <i>Accepted</i> - setVariableResult[1].attributeStatus must be <i>Accepted</i> <p>* Step 16:</p> <p>Message: SetChargingProfileResponse</p> <ul style="list-style-type: none"> - status must be <i>Rejected</i> - statusInfo.reasonCode must be <i>MissingDevModelInfo</i> <p>* Step 18:</p> <p>Message: SetVariablesResponse</p> <ul style="list-style-type: none"> - setVariableResult[0].attributeStatus must be <i>Accepted</i> - setVariableResult[1].attributeStatus must be <i>Accepted</i> <p>* Step 20:</p> <p>Message: SetChargingProfileResponse</p> <ul style="list-style-type: none"> - status must be <i>Rejected</i> - statusInfo.reasonCode must be <i>MissingDevModelInfo</i> <p>* Step 22:</p> <p>Message: SetVariablesResponse</p> <ul style="list-style-type: none"> - setVariableResult[0].attributeStatus must be <i>Accepted</i> - setVariableResult[1].attributeStatus must be <i>Accepted</i> <p>* Step 24:</p> <p>Message: SetChargingProfileResponse</p> <ul style="list-style-type: none"> - status must be <i>Accepted</i>
<p>Post scenario validations:</p> <p>N/a</p>

TC_Q_123_CS: Local V2X control for load balancing - not supported

Test case name	Local V2X control for load balancing - not supported
Test case Id	TC_Q_123_CS
Use case Id(s)	Q09
Requirement(s)	Q09.FR.01
System under test	Charging Station
Description	To allow the EV to be utilized for locally controlled load balancing.
Purpose	To verify if the Charging Station is able to respond correctly if it doesn't support Local Load Balancing.
Prerequisite(s)	<ul style="list-style-type: none"> - EV supports V2X power transfer loop - EV and charging station support ISO15118-20 or CHAdeMO - V2XChargingCtrlr.SupportedOperationModes doesn't contain <i>LocalLoadBalancing</i>

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): IF <Configured EVProtocol> is <i>CHAdeMO</i> THEN State is <i>Authorized</i> State is <i>EVConnectedPreSession</i> ELSE IF <Configured EVProtocol> is <i>ISO15118</i> THEN State is <i>Iso1511820V2xControlBpt</i> State is <i>EVConnectedPreSession</i> State is <i>Authorized15118</i> (for 15118-20) END IF

Main (Test scenario)	
Charging Station	CSMS
Send LocalLoadBalancing profile	
2. The Charging Station responds with a SetChargingProfileResponse	1. The Test System sends a SetChargingProfileRequest with evseld <Configured evseld> chargingProfile - transactionId <transactionId> - chargingProfilePurpose TxProfile - chargingSchedule.chargingRateUnit W - chargingSchedule.chargingSchedulePeriod.operationMode LocalLoadBalancing

Tool validations
* Step 2: Message: SetChargingProfileResponse - status must be <i>Rejected</i> - statusInfo.reasonCode must be <i>UnsupportedParam</i>
Post scenario validations: N/a

TC_Q_125_CS: Idle operationMode - Idle, minimizing energy consumption - Idle with EvseSleep

Test case name	Idle operationMode - Idle, minimizing energy consumption - Idle with EvseSleep
Test case Id	TC_Q_125_CS
Use case Id(s)	Q10
Requirement(s)	Q10.FR.01, Q10.FR.03, Q10.FR.04
System under test	Charging Station
Description	To request the EV to not perform any charging or discharging. Preconditioning of the vehicle is allowed.
Purpose	To verify if the Charging Station is able to stop charging in Idle and report evseSleep.
Prerequisite(s)	<ul style="list-style-type: none"> - EV supports V2X power transfer loop - EV and charging station support ISO15118-20 or CHAdeMO - V2XChargingCtrlr.SupportedOperationModes contains <i>Idle</i> - SmartChargingCtrlr.SupportsFeature[EvseSleep] is <i>true</i>

Before (Preparations)
Configuration State: - SampledDataCtrlr.TxUpdatedInterval = 10 - SampledDataCtrlr.TxUpdatedMeasurands = <i>Energy.Active.Import.Register</i>
Memory State: N/a
Reusable State(s): State is <i>EVConnectedPreSession</i> IF <Configured EVProtocol> is <i>CHAdeMO</i> THEN - State is <i>Authorized</i> ELSE - State is <i>Iso1511820V2xControlBpt</i> END IF State <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
Without EVSE sleeping	
2. The Charging Station responds with a SetChargingProfileResponse with status must be <i>Accepted</i>	1. The Test System sends a SetChargingProfileRequest with evseld <i><Configured evseld></i> chargingProfile - transactionId <i><transactionId></i> - chargingProfilePurpose <i>TxProfile</i> - chargingSchedule.chargingRateUnit <i>W</i> - chargingSchedule.chargingSchedulePeriod.operationMode <i>Idle</i> - chargingSchedule.chargingSchedulePeriod.evseSleep <i>false</i>
3. The Charging Station sends following TransactionEventRequest	4. The Test System responds with a TransactionEventResponse
5. The Charging Station sends following TransactionEventRequest	6. The Test System responds with a TransactionEventResponse
With EVSE sleeping	

Main (Test scenario)	
8. The Charging Station responds with a SetChargingProfileResponse	7. The Test System sends a SetChargingProfileRequest with evseld <Configured evseld> chargingProfile - transactionId <transactionId> - chargingProfilePurpose TxProfile - chargingSchedule.chargingRateUnit W - chargingSchedule.chargingSchedulePeriod.operationMode Idle - chargingSchedule.chargingSchedulePeriod.evseSleep true
9. The Charging Station sends following TransactionEventRequest	10. The Test System responds with a TransactionEventResponse
11. The Charging Station sends following TransactionEventRequest	12. The Test System responds with a TransactionEventResponse

Tool validations
<p>* Step 2: Message: SetChargingProfileResponse - status must be <i>Accepted</i></p> <p>* Step 3: Message: TransactionEventRequest - eventType must be <i>Updated</i> - transactionInfo.transactionId must be <transactionId> - evseSleep must be <i>false</i> <omitted> - meterValue[0].sampledValue[0].measurand must be <i>Energy.Active.Import.Register</i></p> <p>* Step 5: Message: TransactionEventRequest - eventType must be <i>Updated</i> - transactionInfo.transactionId must be <transactionId> - evseSleep must be <i>false</i> <omitted></p> <p>* Step 8: Message: SetChargingProfileResponse - status must be <i>Accepted</i></p> <p>* Step 9: Message: TransactionEventRequest - eventType must be <i>Updated</i> - eventData.transactionData.transactionId must be <transactionId> - evseSleep must be <i>true</i> - meterValue[0].sampledValue[0].measurand must be <i>Energy.Active.Import.Register</i></p> <p>* Step 11: Message: TransactionEventRequest - eventType must be <i>Updated</i> - eventData.transactionData.transactionId must be <transactionId> - evseSleep must be <i>true</i> - meterValue[0].sampledValue[0].measurand must be <i>Energy.Active.Import.Register</i> - meterValue[0].sampledValue[0].value must be identical to value in Step 9</p> <p>Post scenario validations: N/a</p>

TC_Q_126_CS: Idle operationMode - Idle, minimizing energy consumption - Idle with EvseSleep unsupported

Test case name	Idle operationMode - Idle, minimizing energy consumption - Idle with EvseSleep unsupported
Test case Id	TC_Q_126_CS
Use case Id(s)	Q10
Requirement(s)	Q10.FR.05
System under test	Charging Station
Description	To request the EV to not perform any charging or discharging. Preconditioning of the vehicle is allowed.
Purpose	To verify if the Charging Station reports evseSleep as false when not supported
Prerequisite(s)	<ul style="list-style-type: none"> - EV supports V2X power transfer loop - EV and charging station support ISO15118-20 or CHAdeMO - V2XChargingCtrlr.SupportedOperationModes contains <i>Idle</i> - SmartChargingCtrlr.SupportsFeature[EvseSleep] is <i>false</i> or absent

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>EVConnectedPreSession</i> IF <Configured EVProtocol> is <i>CHAdeMO</i> THEN - State is <i>Authorized</i> ELSE - State is <i>Iso1511820V2xControlBpt</i> END IF State <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
<i>Profile with EVSE sleep</i>	
2. The Charging Station responds with a SetChargingProfileResponse	1. The Test System sends a SetChargingProfileRequest with evseld <Configured evseld> chargingProfile - transactionId <transactionId> - chargingProfilePurpose <i>TxProfile</i> - chargingSchedule.chargingRateUnit <i>W</i> - chargingSchedule.chargingSchedulePeriod.operationMode <i>Idle</i> - chargingSchedule.chargingSchedulePeriod.evseSleep <i>true</i>
3. The Charging Station sends following TransactionEventRequest	4. The Test System responds with a TransactionEventResponse

Tool validations
* Step 2: Message: SetChargingProfileResponse - status must be <i>Accepted</i> * Step 3: Message: TransactionEventRequest - eventType must be <i>Updated</i> - eventData.transactionData.transactionId must be <transactionId> - evseSleep must be <i>false</i> <omitted>

Tool validations
Post scenario validations: N/a

TC_Q_127_CS: Idle operationMode - Idle, minimizing energy consumption - Charging profile validations

Test case name	Idle operationMode - Idle, minimizing energy consumption - Charging profile validations
Test case Id	TC_Q_127_CS
Use case Id(s)	Q10
Requirement(s)	Q10.FR.02
System under test	Charging Station
Description	To request the EV to not perform any charging or discharging. Preconditioning of the vehicle is allowed.
Purpose	To verify if the Charging Station rejects charging profiles with value for limit,dischargingLimit and setpoints.
Prerequisite(s)	<ul style="list-style-type: none"> - EV supports V2X power transfer loop - EV and charging station support ISO15118-20 or CHAdeMO - V2XChargingCtrlr.SupportedOperationModes contains <i>Idle</i>

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>EVConnectedPreSession</i> IF <Configured EVProtocol> is <i>CHAdeMO</i> THEN - State is <i>Authorized</i> ELSE - State is <i>Iso1511820V2xControlBpt</i> END IF

Main (Test scenario)	
Charging Station	CSMS
Limits specified	

Main (Test scenario)	
2. The Charging Station responds with a SetChargingProfileResponse	<div>1. The Test System sends a SetChargingProfileRequest with evseld <Configured evseld></div> <div>chargingProfile</div> <div>- transactionId <transactionId></div> <div>- chargingProfilePurpose TxProfile</div> <div>- chargingSchedule.chargingRateUnit W</div> <div>- chargingSchedule.chargingSchedulePeriod.operationMode Idle</div> <div>- chargingSchedule.chargingSchedulePeriod.limit <not omitted></div> <div>- chargingSchedule.chargingSchedulePeriod.limit_L2 <not omitted></div> <div>- chargingSchedule.chargingSchedulePeriod.limit_L3 <not omitted></div> <div>- chargingSchedule.chargingSchedulePeriod.dischargeLimit <omitted></div> <div>- chargingSchedule.chargingSchedulePeriod.dischargeLimit_L2 <omitted></div> <div>- chargingSchedule.chargingSchedulePeriod.dischargeLimit_L3 <omitted></div> <div>- chargingSchedule.chargingSchedulePeriod.setpoint <omitted></div> <div>- chargingSchedule.chargingSchedulePeriod.setpoint_L2 <omitted></div> <div>- chargingSchedule.chargingSchedulePeriod.setpoint_L3 <omitted></div> <div>- chargingSchedule.chargingSchedulePeriod.setpointReactive <omitted></div> <div>- chargingSchedule.chargingSchedulePeriod.setpointReactive_L2 <omitted></div> <div>- chargingSchedule.chargingSchedulePeriod.setpointReactive_L3 <omitted></div>
DischargingLimits specified	

Main (Test scenario)	
4. The Charging Station responds with a SetChargingProfileResponse	<div>3. The Test System sends a SetChargingProfileRequest with evseld <Configured evseld></div> <div>chargingProfile</div> <div>- transactionId <transactionId></div> <div>- chargingProfilePurpose TxProfile</div> <div>- chargingSchedule.chargingRateUnit W</div> <div>- chargingSchedule.chargingSchedulePeriod.operationMode Idle</div> <div>- chargingSchedule.chargingSchedulePeriod.limit <omitted></div> <div>- chargingSchedule.chargingSchedulePeriod.limit_L2 <omitted></div> <div>- chargingSchedule.chargingSchedulePeriod.limit_L3 <omitted></div> <div>- chargingSchedule.chargingSchedulePeriod.dischargeLimit <not omitted></div> <div>- chargingSchedule.chargingSchedulePeriod.dischargeLimit_L2 <not omitted></div> <div>- chargingSchedule.chargingSchedulePeriod.dischargeLimit_L3 <not omitted></div> <div>- chargingSchedule.chargingSchedulePeriod.setpoint <omitted></div> <div>- chargingSchedule.chargingSchedulePeriod.setpoint_L2 <omitted></div> <div>- chargingSchedule.chargingSchedulePeriod.setpoint_L3 <omitted></div> <div>- chargingSchedule.chargingSchedulePeriod.setpointReactive <omitted></div> <div>- chargingSchedule.chargingSchedulePeriod.setpointReactive_L2 <omitted></div> <div>- chargingSchedule.chargingSchedulePeriod.setpointReactive_L3 <omitted></div>
Setpoints specified	

Main (Test scenario)	
6. The Charging Station responds with a SetChargingProfileResponse	<div>5. The Test System sends a SetChargingProfileRequest with evseld <Configured evseld></div> <div>chargingProfile</div> <div>- transactionId <transactionId></div> <div>- chargingProfilePurpose TxProfile</div> <div>- chargingSchedule.chargingRateUnit W</div> <div>- chargingSchedule.chargingSchedulePeriod.operationMode Idle</div> <div>- chargingSchedule.chargingSchedulePeriod.limit <omitted></div> <div>- chargingSchedule.chargingSchedulePeriod.limit_L2 <omitted></div> <div>- chargingSchedule.chargingSchedulePeriod.limit_L3 <omitted></div> <div>- chargingSchedule.chargingSchedulePeriod.dischargeLimit <omitted></div> <div>- chargingSchedule.chargingSchedulePeriod.dischargeLimit_L2 <omitted></div> <div>- chargingSchedule.chargingSchedulePeriod.dischargeLimit_L3 <omitted></div> <div>- chargingSchedule.chargingSchedulePeriod.setpoint <not omitted></div> <div>- chargingSchedule.chargingSchedulePeriod.setpoint_L2 <not omitted></div> <div>- chargingSchedule.chargingSchedulePeriod.setpoint_L3 <not omitted></div> <div>- chargingSchedule.chargingSchedulePeriod.setpointReactive <omitted></div> <div>- chargingSchedule.chargingSchedulePeriod.setpointReactive_L2 <omitted></div> <div>- chargingSchedule.chargingSchedulePeriod.setpointReactive_L3 <omitted></div>
SetpointReactive specified	

Main (Test scenario)	
8. The Charging Station responds with a SetChargingProfileResponse	<div>7. The Test System sends a SetChargingProfileRequest with evseld <Configured evseld></div> <div>chargingProfile</div> <div>- transactionId <transactionId></div> <div>- chargingProfilePurpose TxProfile</div> <div>- chargingSchedule.chargingRateUnit W</div> <div>- chargingSchedule.chargingSchedulePeriod.operationMode Idle</div> <div>- chargingSchedule.chargingSchedulePeriod.limit <omitted></div> <div>- chargingSchedule.chargingSchedulePeriod.limit_L2 <omitted></div> <div>- chargingSchedule.chargingSchedulePeriod.limit_L3 <omitted></div> <div>- chargingSchedule.chargingSchedulePeriod.dischargeLimit <omitted></div> <div>- chargingSchedule.chargingSchedulePeriod.dischargeLimit_L2 <omitted></div> <div>- chargingSchedule.chargingSchedulePeriod.dischargeLimit_L3 <omitted></div> <div>- chargingSchedule.chargingSchedulePeriod.setpoint <omitted></div> <div>- chargingSchedule.chargingSchedulePeriod.setpoint_L2 <omitted></div> <div>- chargingSchedule.chargingSchedulePeriod.setpoint_L3 <omitted></div> <div>- chargingSchedule.chargingSchedulePeriod.setpointReactive <not omitted></div> <div>- chargingSchedule.chargingSchedulePeriod.setpointReactive_L2 <not omitted></div> <div>- chargingSchedule.chargingSchedulePeriod.setpointReactive_L3 <not omitted></div>
All ok	

Main (Test scenario)	
<p>10. The Charging Station responds with a SetChargingProfileResponse</p>	<p>9. The Test System sends a SetChargingProfileRequest with evseld <Configured evseld></p> <p>chargingProfile</p> <ul style="list-style-type: none"> - transactionId <transactionId> - chargingProfilePurpose TxProfile - chargingSchedule.chargingRateUnit W - chargingSchedule.chargingSchedulePeriod.operationMode Idle - chargingSchedule.chargingSchedulePeriod.limit <omitted> - chargingSchedule.chargingSchedulePeriod.limit_L2 <omitted> - chargingSchedule.chargingSchedulePeriod.limit_L3 <omitted> - chargingSchedule.chargingSchedulePeriod.dischargeLimit <omitted> - chargingSchedule.chargingSchedulePeriod.dischargeLimit_L2 <omitted> - chargingSchedule.chargingSchedulePeriod.dischargeLimit_L3 <omitted> - chargingSchedule.chargingSchedulePeriod.setpoint <omitted> - chargingSchedule.chargingSchedulePeriod.setpoint_L2 <omitted> - chargingSchedule.chargingSchedulePeriod.setpoint_L3 <omitted> - chargingSchedule.chargingSchedulePeriod.setpointReactive <omitted> - chargingSchedule.chargingSchedulePeriod.setpointReactive_L2 <omitted> - chargingSchedule.chargingSchedulePeriod.setpointReactive_L3 <omitted>

Tool validations
<p>* Step 2:</p> <p>Message: SetChargingProfileResponse</p> <ul style="list-style-type: none"> - status must be <i>Rejected</i> - statusInfo.reasonCode must be <i>InvalidSchedule</i> <p>* Step 4:</p> <p>Message: SetChargingProfileResponse</p> <ul style="list-style-type: none"> - status must be <i>Rejected</i> - statusInfo.reasonCode must be <i>InvalidSchedule</i> <p>* Step 6:</p> <p>Message: SetChargingProfileResponse</p> <ul style="list-style-type: none"> - status must be <i>Rejected</i> - statusInfo.reasonCode must be <i>InvalidSchedule</i> <p>* Step 8:</p> <p>Message: SetChargingProfileResponse</p> <ul style="list-style-type: none"> - status must be <i>Rejected</i> - statusInfo.reasonCode must be <i>InvalidSchedule</i> <p>* Step 10:</p> <p>Message: SetChargingProfileResponse</p> <ul style="list-style-type: none"> - status must be <i>Accepted</i> <p>Post scenario validations:</p> <p>N/a</p>

TC_Q_128_CS: Going offline during V2X operation - invalidAfterOfflineDuration = true

Test case name	Going offline during V2X operation - invalidAfterOfflineDuration = true
Test case Id	TC_Q_128_CS
Use case Id(s)	Q11, K01
Requirement(s)	K01.FR.101, K01.FR.102, K01.FR.103
System under test	Charging Station
Description	To describe the amount of time that V2X operations may continue when the Charging Station is offline.
Purpose	To verify if the Charging station stops using the charging profile when Charging station is longer offline than specified in the charging profile's <code>maxOfflineDuration</code> .
Prerequisite(s)	- EV and charging station support ISO15118-20 or CHAdeMO

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State: IF <Configured EVProtocol> is <i>CHAdeMO</i> THEN State is <i>Authorized</i> State is <i>EVConnectedPreSession</i> ELSE IF <Configured EVProtocol> is <i>ISO15118</i> THEN State is <i>Iso1511820V2xControlBpt</i> State is <i>EVConnectedPreSession</i> State is <i>Authorized15118</i> (for 15118-20) END IF

Main (Test scenario)	
Charging Station	CSMS
1. The Charging Station sends a NotifyEVChargingNeedsRequest <u>Note(s):</u> - This step is applicable if <Configured EVProtocol> is <i>ISO15118</i> and is captured by <i>Authorized15118</i> (for 15118-20), but validated on testcase level.	2. Test System responds with a NotifyEVChargingNeedsResponse with status <i>Accepted</i>
4. The Charging Station responds with a SetChargingProfileResponse	3. The Test System sends a SetChargingProfileRequest with evseld <i>0</i> chargingProfile.id <Configured chargingProfileId> chargingProfile.stackLevel <i>0</i> chargingProfile.chargingProfilePurpose <i>TxDefaultProfile</i> chargingProfile.chargingSchedule.chargingRateUnit <Configured chargingRateUnit> chargingProfile.chargingSchedule.chargingSchedulePeriod[0].startPeriod <i>0</i> chargingProfile.chargingSchedule.chargingSchedulePeriod[0].startPoint <i>-6 * <limit multiplier></i> Note: Check <i>Determine Charging Profile Limit Multiplier</i> for <limit multiplier> chargingProfile.chargingSchedule.chargingSchedulePeriod[0].numberOfPhases <Configured numberPhases>

Main (Test scenario)	
6. The Charging Station responds with a SetChargingProfileResponse	<p>5. The Test System sends a SetChargingProfileRequest with evseld 0</p> <p>chargingProfile.id <Configured chargingProfileId2></p> <p>chargingProfile.stackLevel 1</p> <p>chargingProfile.chargingProfilePurpose TxDefaultProfile</p> <p>chargingProfile.maxOfflineDuration <Configured maxOfflineDuration></p> <p>chargingProfile.invalidAfterOfflineDuration true</p> <p>chargingProfile.chargingSchedule.chargingRateUnit <Configured chargingRateUnit></p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod[0].startPeriod 0</p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod[0].startPoint -7 * <limit multiplier></p> <p>Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier></p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod[0].numberPhases <Configured numberPhases></p>
<p>7. The Charging station sends a NotifyEVChargingScheduleRequest</p> <p>Note(s): - This step is applicable if <Configured EVProtocol> is ISO15118</p>	<p>8. The Test System responds with a NotifyEVChargingScheduleResponse with status Accepted</p>
9. Execute Reusable State EnergyTransferStarted	
10. The Test System closes the WebSocket connection AND does not accept a reconnect.	
11. The Test System waits for the duration of <Configured maxOfflineDuration> - 20 seconds.	
12. The Test System accepts reconnection attempt from the Charging Station.	
14. The Charging Station responds with a GetCompositeScheduleResponse	<p>13. The Test System sends a GetCompositeScheduleRequest with evseld <Configured evseld></p> <p>duration is 900</p> <p>chargingRateUnit <Configured chargingRateUnit></p>
15. The Test System closes the WebSocket connection AND does not accept a reconnect.	
16. The Test System waits for the duration of <Configured maxOfflineDuration> + 20.	
17. The Test System accepts reconnection attempt from the Charging Station.	
19. The Charging Station responds with a GetCompositeScheduleResponse	<p>18. The Test System sends a GetCompositeScheduleRequest with evseld <Configured evseld></p> <p>duration is 900</p> <p>chargingRateUnit <Configured chargingRateUnit></p>

Main (Test scenario)	
21. The Charging Station responds with a SetChargingProfileResponse	<p>20. The Test System sends a SetChargingProfileRequest with evseld 0 chargingProfile.id <Configured chargingProfileId2> chargingProfile.stackLevel 1 chargingProfile.chargingProfilePurpose TxDefaultProfile chargingProfile.maxOfflineDuration <Configured maxOfflineDuration> chargingProfile.invalidAfterOfflineDuration false chargingProfile.chargingSchedule.chargingRateUnit <Configured chargingRateUnit> chargingProfile.chargingSchedule.chargingSchedulePeriod[0].startPeriod 0 chargingProfile.chargingSchedule.chargingSchedulePeriod[0].startPoint -8 * <limit multiplier> Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier> chargingProfile.chargingSchedule.chargingSchedulePeriod[0].numberPhases <Configured numberPhases></p>
22. The Test System closes the WebSocket connection AND does not accept a reconnect.	
23. The Test System waits for the duration of <Configured maxOfflineDuration> + 20.	
24. The Test System accepts reconnection attempt from the Charging Station.	
26. The Charging Station responds with a GetCompositeScheduleResponse	<p>25. The Test System sends a GetCompositeScheduleRequest with evseld <Configured evseld> duration is 900 chargingRateUnit <Configured chargingRateUnit></p>
28. The Charging Station responds with a SetChargingProfileResponse	<p>27. The Test System sends a SetChargingProfileRequest with evseld 0 chargingProfile.id <Configured chargingProfileId2> chargingProfile.stackLevel 1 chargingProfile.chargingProfilePurpose TxDefaultProfile chargingProfile.maxOfflineDuration is absent chargingProfile.invalidAfterOfflineDuration is absent chargingProfile.chargingSchedule.chargingRateUnit <Configured chargingRateUnit> chargingProfile.chargingSchedule.chargingSchedulePeriod[0].startPeriod 0 chargingProfile.chargingSchedule.chargingSchedulePeriod[0].startPoint -9 * <limit multiplier> Note: Check Determine Charging Profile Limit Multiplier for <limit multiplier> chargingProfile.chargingSchedule.chargingSchedulePeriod[0].numberPhases <Configured numberPhases></p>
29. The Test System closes the WebSocket connection AND does not accept a reconnect.	
30. The Test System waits for the duration of <Configured maxOfflineDuration> + 20.	
31. The Test System accepts reconnection attempt from the Charging Station.	
33. The Charging Station responds with a GetCompositeScheduleResponse	<p>32. The Test System sends a GetCompositeScheduleRequest with evseld <Configured evseld> duration is 900 chargingRateUnit <Configured chargingRateUnit></p>

Tool validations

* Step 1:

Message **NotifyEVChargingNeedsRequest**

- **evseld** = <Configured evseld>

For AC charging station:

- **chargingNeeds.requestedEnergyTransfer** AC_BPT

- **chargingNeeds.availableEnergyTransfer** [AC_single_phase, AC_two_phase, AC_three_phase, AC_BPT, AC_BPT_DER, AC_DER]

For DC charging station:

- **chargingNeeds.requestedEnergyTransfer** DC_BPT

- **chargingNeeds.availableEnergyTransfer** [DC, DC_BPT, DC_ACDP, DC_ACDP_BPT]

- **chargingNeeds.controlMode** = DynamicControl

- **chargingNeeds.v2xChargingParameters.maxChargePower** = 10000 (W)

- **chargingNeeds.v2xChargingParameters.maxDischargePower** = 5000 (W)

* Step 4:

Message **SetChargingProfileResponse**

- **status** must be Accepted

* Step 6:

Message **SetChargingProfileResponse**

- **status** must be Accepted

* Step 14:

Message: **GetCompositeScheduleResponse**

- **status** must be Accepted

- **schedule.chargingRateUnit** must be <Configured chargingRateUnit>

- **schedule.chargingSchedulePeriod[0].startPeriod** must be 0

- **schedule.chargingSchedulePeriod[0].setpoint** must be -7 * <limit multiplier>

Note: Check [Determine Charging Profile Limit Multiplier](#) for <limit multiplier>

* Step 19:

Message: **GetCompositeScheduleResponse**

- **status** must be Accepted

- **schedule.chargingRateUnit** must be <Configured chargingRateUnit>

- **schedule.chargingSchedulePeriod[0].startPeriod** must be 0

- **schedule.chargingSchedulePeriod[0].setpoint** must be -6 * <limit multiplier>

Note: Check [Determine Charging Profile Limit Multiplier](#) for <limit multiplier>

* Step 26:

Message: **GetCompositeScheduleResponse**

- **status** must be Accepted

- **schedule.chargingRateUnit** must be <Configured chargingRateUnit>

- **schedule.chargingSchedulePeriod[0].startPeriod** must be 0

- **schedule.chargingSchedulePeriod[0].setpoint** must be -8 * <limit multiplier>

Note: Check [Determine Charging Profile Limit Multiplier](#) for <limit multiplier>

* Step 33:

Message: **GetCompositeScheduleResponse**

- **status** must be Accepted

- **schedule.chargingRateUnit** must be <Configured chargingRateUnit>

- **schedule.chargingSchedulePeriod[0].startPeriod** must be 0

- **schedule.chargingSchedulePeriod[0].setpoint** must be -9 * <limit multiplier>

Note: Check [Determine Charging Profile Limit Multiplier](#) for <limit multiplier>

Post scenario validations:

N/a

TC_Q_130_CS: V2X Authorisation - ISO15118-20 - has ISO15118ServiceRenegotiationSupport - Charging needs rejected

Test case name	V2X Authorisation - ISO15118-20 - has ISO15118ServiceRenegotiationSupport - Charging needs rejected
Test case Id	TC_Q_130_CS
Use case Id(s)	Q01
Requirement(s)	Q01.FR.03, Q01.FR.05
System under test	Charging Station
Description	Authorization of an EV by the CSMS to start a V2X power transfer loop.
Purpose	To verify if the Charging Station is able to end the transaction when the charging needs are rejected and it does has ISO15118ServiceRenegotiationSupport.
Prerequisite(s)	<ul style="list-style-type: none"> - EV supports V2X power transfer loop - EV and charging station support ISO15118-20 - Charging Station has ISO15118ServiceRenegotiationSupport (component variable <i>ISO15118Ctrlr.ServiceRenegotiationSupport</i> is true)

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>EVConnectedPreSession</i> State is <i>Authorized15118</i> (for 15118-20) State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
Agree on energy transfer mode	
<u>Manual Action:</u> Use EV to communicate EV charging needs:	
<u>For AC charging station:</u>	
requestedEnergyTransfer <i>AC_BPT</i>	
availableEnergyTransfer [<i>AC_single_phase, AC_two_phase, AC_three_phase, AC_BPT, AC_BPT_DER, AC_DER</i>]	
 <u>For DC charging station:</u>	
requestedEnergyTransfer <i>DC_BPT</i>	
availableEnergyTransfer [<i>DC, DC_BPT, DC_ACDP, DC_ACDP_BPT</i>]	
 controlMode <i>ScheduledControl</i>	
maxChargePower <i>10000 (W)</i>	
maxDischargePower <i>5000 (W)</i>	
1. The Charging Station sends a NotifyEVChargingNeedsRequest	2. The Test System responds with a NotifyEVChargingNeedsResponse with - status <i>Rejected</i>
Renegotiate	

Main (Test scenario)	
<p><u>Manual Action:</u> Use EV to communicate EV charging needs:</p> <p>For AC charging station:</p> <p>requestedEnergyTransfer <i>AC_two_phase</i></p> <p>availableEnergyTransfer [<i>AC_single_phase, AC_two_phase, AC_three_phase, AC_BPT, AC_BPT_DER, AC_DER</i>]</p> <p>For DC charging station:</p> <p>requestedEnergyTransfer <i>DC</i></p> <p>availableEnergyTransfer [<i>DC, DC_BPT, DC_ACDP, DC_ACDP_BPT</i>]</p> <p>controlMode <i>ScheduledControl</i></p> <p>maxChargePower <i>10000 (W)</i></p>	
<p>3. The Charging Station sends a NotifyEVChargingNeedsRequest</p>	<p>4. The Test System responds with a NotifyEVChargingNeedsResponse with</p> <p>- status <i>NoChargingProfile</i></p>

Tool validations
<p>* Step 1:</p> <p>Message: NotifyEVChargingNeedsRequest</p> <p>- evseld must be <i><Configured evseld></i></p> <p>For AC charging station:</p> <p>- chargingNeeds.requestedEnergyTransfer <i>AC_BPT</i></p> <p>- chargingNeeds.availableEnergyTransfer [<i>AC_single_phase, AC_two_phase, AC_three_phase, AC_BPT, AC_BPT_DER, AC_DER</i>]</p> <p>For DC charging station:</p> <p>- chargingNeeds.requestedEnergyTransfer <i>DC_BPT</i></p> <p>- chargingNeeds.availableEnergyTransfer [<i>DC, DC_BPT, DC_ACDP, DC_ACDP_BPT</i>]</p> <p>- chargingNeeds.controlMode must be <i>ScheduledControl</i></p> <p>- chargingNeeds.v2xChargingParameters.maxChargePower must be <i>10000 (W)</i></p> <p>- chargingNeeds.v2xChargingParameters.maxDischargePower must be <i>5000 (W)</i></p> <p>* Step 3:</p> <p>Message: NotifyEVChargingNeedsRequest</p> <p>- evseld must be <i><Configured evseld></i></p> <p>For AC charging station:</p> <p>- chargingNeeds.requestedEnergyTransfer <i>AC_two_phase</i></p> <p>- chargingNeeds.availableEnergyTransfer [<i>AC_single_phase, AC_two_phase, AC_three_phase, AC_BPT, AC_BPT_DER, AC_DER</i>]</p> <p>For DC charging station:</p> <p>- chargingNeeds.requestedEnergyTransfer <i>DC</i></p> <p>- chargingNeeds.availableEnergyTransfer [<i>DC, DC_BPT, DC_ACDP, DC_ACDP_BPT</i>]</p> <p>- chargingNeeds.controlMode must be <i>ScheduledControl</i></p> <p>- chargingNeeds.v2xChargingParameters.maxChargePower must be <i>10000 (W)</i></p> <p>- chargingNeeds.v2xChargingParameters.maxDischargePower must be <i><omitted></i></p> <p>Post scenario validations:</p> <p>N/a</p>

R DER Control

TC_R_100_CS: Configure DER control - Persistent DERControls

Test case name	Starting a V2X session with DER control in EVSE - Persistent DERControls
Test case Id	TC_R_100_CS
Use case Id(s)	R01
Requirement(s)	R01.FR.02
System under test	Charging Station
Description	Setting some DER controls and reboot the charging station. After reboot, retrieving the configured DER controls.
Purpose	To check if the configured DER Controls are persistent.
Prerequisite(s)	CS supports DER Control

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 sends a ClearDERControlResponse	1. The Test System sends a ClearDERControlRequest with isDefault true
4. The Charging Station sends a ClearDERControlResponse	3. The Test System sends a ClearDERControlRequest with isDefault false
6. The Charging Station sends a SetDERControlResponse	5. The Test System sends a SetDERControlRequest with controlId control1 isDefault true controlType <CS supported controlType1> *.priority 6
8. The Charging Station sends a SetDERControlResponse	7. The Test System sends a SetDERControlRequest with controlId control2 isDefault false controlType <CS supported controlType2> *.priority 6 *.startTime <Current DateTime + 24 hours> *.duration 300
10. The Charging Station responds with a ResetResponse	9. The Test System sends a ResetRequest with type Immediate
12. The Charging Station sends a GetDERControlResponse	11. The Test System sends a GetDERControlRequest with isDefault true
13. The Charging Station sends a ReportDERControlRequest	14 The Test System sends a ReportDERControlResponse
Note(s): - If tbc is True at Step 13 then step 13 and 14 will be repeated	
16. The Charging Station sends a GetDERControlResponse	15. The Test System sends a GetDERControlRequest with isDefault false
17. The Charging Station sends a ReportDERControlRequest	18 The Test System sends a ReportDERControlResponse

Main (Test scenario)Note(s):

- If **tbc** is True at Step 17 then step 17 and 18 will be repeated

Tool validations

* Step 2:

Message: **ClearDERControlResponse**

- **status** must be *Accepted*

* Step 4:

Message: **ClearDERControlResponse**

- **status** must be *Accepted*

* Step 6:

Message: **SetDERControlResponse**

- **status** must be *Accepted*

* Step 8:

Message: **SetDERControlResponse**

- **status** must be *Accepted*

* Step 10:

Message: **ResetResponse**

- **status** must be *Accepted*

* Step 12:

Message: **GetDERControlResponse**

- **status** must be *Accepted*

* Step 13:

Message: **ReportDERControlRequest**

Field corresponding to configuring <CS supported controlType1>

- ***[0].id** must be *control1*

Field corresponding to configuring <CS supported controlType2>

- *Corresponding field should be omitted.*

* Step 16:

Message: **GetDERControlResponse**

- **status** must be *Accepted*

* Step 17:

Message: **ReportDERControlRequest**

Field corresponding to configuring <CS supported controlType2>

- ***[0].id** must be *control2*

Field corresponding to configuring <CS supported controlType1>

- *Corresponding field should be omitted.*

Post scenario validations:

N/a

TC_R_101_CS: Configure DER control - Device model

Test case name	Starting a V2X session with DER control in EVSE - Device model
Test case Id	TC_R_101_CS
Use case Id(s)	R01
Requirement(s)	R01.FR.01
System under test	Charging Station
Description	Retrieve custom report from CS to check if the CS reports the mandatory configuration variables for component DCDERCtrlr.
Purpose	To check if the CS reports the mandatory configuration variables for component DCDERCtrlr.
Prerequisite(s)	CS supports DER Control CS is a DC charging station

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: GetVariablesResponse	1. Test System sends GetVariablesRequest with: component <i>DCDERCtrlr</i> variable <i>MaxW</i>
<i>Step 1 and 2 are repeated as often as needed to report all configuration variables from below list:</i>	
<ul style="list-style-type: none"> - OverExcitedW - OverExcitedPF - UnderExcitedW - UnderExcitedPF - MaxVA - MaxVar - MaxVarNeg - MaxChargeRateW - MaxChargeRateVA - ModesSupported - InverterManufacturer - InverterModel - InverterSwVersion - InverterHwVersion - ReactiveSusceptance 	

Tool validations

* Step 2:

For each message: **GetVariablesRespon**s

- **attributeStatus** must be *Accepted*
- **getVariableResult[0].component.name** must be *"DCDERCtrlr"*
- **getVariableResult[0].variable.name** must be *"MaxW"*
- **getVariableResult[0].attributeType** must be *Actual*
- **getVariableResult[0].attributeValue** must be *<not omitted>*

- **getVariableResult[0].component.name** must be *"DCDERCtrlr"*
- **getVariableResult[0].variable.name** must be *"MaxW"*
- **getVariableResult[0].attributeType** must be *Actual*
- **getVariableResult[0].attributeValue** must be *<not omitted>*

- **getVariableResult[0].component.name** must be *"DCDERCtrlr"*
- **getVariableResult[0].variable.name** must be *"OverExcitedW"*
- **getVariableResult[0].attributeType** must be *Actual*
- **getVariableResult[0].attributeValue** must be *<not omitted>*

- **getVariableResult[0].component.name** must be *"DCDERCtrlr"*
- **getVariableResult[0].variable.name** must be *"OverExcitedPF"*
- **getVariableResult[0].attributeType** must be *Actual*
- **getVariableResult[0].attributeValue** must be *<not omitted>*

- **getVariableResult[0].component.name** must be *"DCDERCtrlr"*
- **getVariableResult[0].variable.name** must be *"UnderExcitedW"*
- **getVariableResult[0].attributeType** must be *Actual*
- **getVariableResult[0].attributeValue** must be *<not omitted>*

- **getVariableResult[0].component.name** must be *"DCDERCtrlr"*
- **getVariableResult[0].variable.name** must be *"UnderExcitedPF"*
- **getVariableResult[0].attributeType** must be *Actual*
- **getVariableResult[0].attributeValue** must be *<not omitted>*

- **getVariableResult[0].component.name** must be *"DCDERCtrlr"*
- **getVariableResult[0].variable.name** must be *"MaxVA"*
- **getVariableResult[0].attributeType** must be *Actual*
- **getVariableResult[0].attributeValue** must be *<not omitted>*

Tool validations

- **getVariableResult[0].component.name** must be "DCDERCtrlr"
 - **getVariableResult[0].variable.name** must be "MaxVar"
 - **getVariableResult[0].attributeType** must be *Actual*
 - **getVariableResult[0].attributeValue** must be <not omitted>
-
- **getVariableResult[0].component.name** must be "DCDERCtrlr"
 - **getVariableResult[0].variable.name** must be "MaxVarNeg"
 - **getVariableResult[0].attributeType** must be *Actual*
 - **getVariableResult[0].attributeValue** must be <not omitted>
-
- **getVariableResult[0].component.name** must be "DCDERCtrlr"
 - **getVariableResult[0].variable.name** must be "MaxChargeRateW"
 - **getVariableResult[0].attributeType** must be *Actual*
 - **getVariableResult[0].attributeValue** must be <not omitted>
-
- **getVariableResult[0].component.name** must be "DCDERCtrlr"
 - **getVariableResult[0].variable.name** must be "MaxChargeRateVA"
 - **getVariableResult[0].attributeType** must be *Actual*
 - **getVariableResult[0].attributeValue** must be <not omitted>
-
- **getVariableResult[0].component.name** must be "DCDERCtrlr"
 - **getVariableResult[0].variable.name** must be "ModesSupported"
 - **getVariableResult[0].attributeType** must be *Actual*
 - **getVariableResult[0].attributeValue** must be <not omitted>
-
- **getVariableResult[0].component.name** must be "DCDERCtrlr"
 - **getVariableResult[0].variable.name** must be "InverterManufacturer"
 - **getVariableResult[0].attributeType** must be *Actual*
 - **getVariableResult[0].attributeValue** must be <not omitted>
-
- **getVariableResult[0].component.name** must be "DCDERCtrlr"
 - **getVariableResult[0].variable.name** must be "InverterModel"
 - **getVariableResult[0].attributeType** must be *Actual*
 - **getVariableResult[0].attributeValue** must be <not omitted>

Tool validations
<ul style="list-style-type: none">- getVariableResult[0].component.name must be <i>"DCDERCtrlr"</i>- getVariableResult[0].variable.name must be <i>"InverterSwVersion"</i>- getVariableResult[0].attributeType must be <i>Actual</i>- getVariableResult[0].attributeValue must be <i><not omitted></i>
<ul style="list-style-type: none">- getVariableResult[0].component.name must be <i>"DCDERCtrlr"</i>- getVariableResult[0].variable.name must be <i>"InverterHwVersion"</i>- getVariableResult[0].attributeType must be <i>Actual</i>- getVariableResult[0].attributeValue must be <i><not omitted></i>
<ul style="list-style-type: none">- getVariableResult[0].component.name must be <i>"DCDERCtrlr"</i>- getVariableResult[0].variable.name must be <i>"ReactiveSusceptance"</i>- getVariableResult[0].attributeType must be <i>Actual</i>- getVariableResult[0].attributeValue must be <i><not omitted></i>
Post scenario validations: N/a

TC_R_102_CS: Configure DER control - Clearing controlTypes

Test case name	Configure DER control settings at CS - clearing controlTypes
Test case Id	TC_R_102_CS
Use case Id(s)	R04
Requirement(s)	R04.FR.02, R04.FR.05, R04.FR.30, R04.FR.37, R04.FR.41, R04.FR.42, R04.FR.44
System under test	Charging Station
Description	Setting some controlTypes and let the Charging Station clear them.
Purpose	To check if the CS behaves correctly when clearing controlTypes in different ways.
Prerequisite(s)	EV supports DER Controls Configured <CS supported controlType1/2> may not be enterService or gradient.

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

Main (Test scenario)	
Charging Station	CSMS
<i>Clearing all controls</i>	
2. The Charging Station sends a ClearDERControlResponse	1. The Test System sends a ClearDERControlRequest with isDefault true
4. The Charging Station sends a ClearDERControlResponse	3. The Test System sends a ClearDERControlRequest with isDefault false
6. The Charging Station sends a GetDERControlResponse	5. The Test System sends a GetDERControlRequest with only a requestId
<i>Installing default control1 and control2</i>	
8. The Charging Station sends a SetDERControlResponse	7. The Test System sends a SetDERControlRequest with controlId control1 isDefault true controlType <CS supported controlType1> *.priority 6
10. The Charging Station sends a SetDERControlResponse	9. The Test System sends a SetDERControlRequest with controlId control2 isDefault true controlType <CS supported controlType2> *.priority 6
<i>Installing scheduled control3 and control4</i>	
12. The Charging Station sends a SetDERControlResponse	11. The Test System sends a SetDERControlRequest with controlId control3 isDefault false controlType <CS supported controlType1> *.priority 6 *.startTime <Current DateTime + 24 hours> *.duration 300

Main (Test scenario)	
14. The Charging Station sends a SetDERControlResponse	13. The Test System sends a SetDERControlRequest with controlId <i>control4</i> isDefault <i>false</i> controlType <i><CS supported controlType2></i> *.priority <i>6</i> *.startTime <i><Current DateTime + 24 hours></i> *.duration <i>300</i>
<i>Clearing default controlType1</i>	
16. The Charging Station sends a ClearDERControlResponse	15. The Test System sends a ClearDERControlRequest with isDefault <i>true</i> controlType <i><CS supported controlType1></i>
18. The Charging Station sends a GetDERControlResponse	17. The Test System sends a GetDERControlRequest with isDefault <i>true</i>
19. The Charging Station sends a ReportDERControlRequest	20. The Test System sends a ReportDERControlResponse
<u>Note(s):</u> <i>- If tbc is True at Step 19 then step 19 and 20 will be repeated</i>	
22. The Charging Station sends a GetDERControlResponse	21. The Test System sends a GetDERControlRequest with isDefault <i>false</i>
23. The Charging Station sends a ReportDERControlRequest	24. The Test System sends a ReportDERControlResponse
<u>Note(s):</u> <i>- If tbc is True at Step 23 then step 23 and 24 will be repeated</i>	
<i>Reinstalling default controlType1</i>	
26. The Charging Station sends a SetDERControlResponse	25. The Test System sends a SetDERControlRequest with controlId <i>control1</i> isDefault <i>true</i> controlType <i><CS supported controlType1></i> *.priority <i>6</i>
<i>Clearing scheduled controlType2</i>	
28. The Charging Station sends a ClearDERControlResponse	27. The Test System sends a ClearDERControlRequest with isDefault <i>false</i> controlType <i><CS supported controlType2></i>
30. The Charging Station sends a GetDERControlResponse	29. The Test System sends a GetDERControlRequest with isDefault <i>true</i>
31. The Charging Station sends a ReportDERControlRequest	32. The Test System sends a ReportDERControlResponse
<u>Note(s):</u> <i>- If tbc is True at Step 31 then step 31 and 32 will be repeated</i>	
34. The Charging Station sends a GetDERControlResponse	33. The Test System sends a GetDERControlRequest with isDefault <i>false</i>
35. The Charging Station sends a ReportDERControlRequest	36. The Test System sends a ReportDERControlResponse
<u>Note(s):</u> <i>- If tbc is True at Step 35 then step 35 and 36 will be repeated</i>	
<i>Reinstalling scheduled control4 with controlType2</i>	

Main (Test scenario)	
38. The Charging Station sends a SetDERControlResponse	37. The Test System sends a SetDERControlRequest with controlId <i>control4</i> isDefault <i>false</i> controlType <i><CS supported controlType2></i> *.priority <i>6</i> *.startTime <i><Current DateTime + 24 hours></i> *.duration <i>300</i>
Clearing default control2	
40. The Charging Station sends a ClearDERControlResponse	39. The Test System sends a ClearDERControlRequest with isDefault <i>true</i> controlId <i>control2</i>
Clearing scheduled control3	
42. The Charging Station sends a ClearDERControlResponse	41. The Test System sends a ClearDERControlRequest with isDefault <i>false</i> controlId <i>control3</i>
44. The Charging Station sends a GetDERControlResponse	43. The Test System sends a GetDERControlRequest with isDefault <i>true</i>
45. The Charging Station sends a ReportDERControlRequest	46. The Test System sends a ReportDERControlResponse
<u>Note(s):</u> - If tbc is True at Step 45 then step 45 and 46 will be repeated	
48. The Charging Station sends a GetDERControlResponse	47. The Test System sends a GetDERControlRequest with isDefault <i>false</i>
49 The Charging Station sends a ReportDERControlRequest	50 The Test System sends a ReportDERControlResponse
<u>Note(s):</u> - If tbc is True at Step 49 then step 49 and 50 will be repeated	

Tool validations

* Step 2:

Message: **ClearDERControlResponse**

- **status** must be *Accepted*

* Step 4:

Message: **ClearDERControlResponse**

- **status** must be *Accepted*

* Step 5: Message: **GetDERControlResponse**

- **status** must be *NotFound*

* Step 8:

Message: **SetDERControlResponse**

- **status** must be *Accepted*

* Step 10:

Message: **SetDERControlResponse**

- **status** must be *Accepted*

* Step 12:

Message: **SetDERControlResponse**

- **status** must be *Accepted*

* Step 14:

Message: **SetDERControlResponse**

- **status** must be *Accepted*

* Step 16:

Message: **ClearDERControlResponse**

- **status** must be *Accepted*

* Step 18: *Get default controls*

Message: **GetDERControlResponse**

- **status** must be *Accepted*

* Step 19:

Message: **ReportDERControlRequest**

- **requestId** matches requestId from GetDERControlRequest of step 17.

Field corresponding to configuring <CS supported controlType1>

- *Corresponding field should be omitted.*

Field corresponding to configuring <CS supported controlType2>

- ***[0].id** must be *control2*

* Step 22: *Get scheduled controls*

Message: **GetDERControlResponse**

- **status** must be *Accepted*

* Step 23:

Message: **ReportDERControlRequest**

- **requestId** matches requestId from GetDERControlRequest of step 21.

Field corresponding to configuring <CS supported controlType1>

- ***[0].id** must be *control3*

Field corresponding to configuring <CS supported controlType2>

- ***[0].id** must be *control4*

Tool validations

* Step 26:

Message: **SetDERControlResponse**

- **status** must be *Accepted*

* Step 28: *Clear scheduled controlType2*

Message: **ClearDERControlResponse**

- **status** must be *NotFound*

* Step 30: *Get default controls*

Message: **GetDERControlResponse**

- **status** must be *Accepted*

* Step 31:

Message: **ReportDERControlRequest**

- **requestId** matches requestId from GetDERControlRequest of step 30.

Field corresponding to configuring <CS supported controlType1>

- ***[0].id** must be *control1*

Field corresponding to configuring <CS supported controlType2>

- ***[0].id** must be *control2*

* Step 34: *Get scheduled controls*

Message: **GetDERControlResponse**

- **status** must be *Accepted*

* Step 35:

Message: **ReportDERControlRequest**

- **requestId** matches requestId from GetDERControlRequest of step 33.

Field corresponding to configuring <CS supported controlType1>

- ***[0].id** must be *control3*

Field corresponding to configuring <CS supported controlType2>

- *Corresponding field should be omitted.*

* Step 38:

Message: **SetDERControlResponse**

- **status** must be *Accepted*

* Step 40:

Message: **ClearDERControlResponse**

- **status** must be *Accepted*

* Step 42:

Message: **ClearDERControlResponse**

- **status** must be *Accepted*

* Step 44: *Get default controls*

Message: **GetDERControlResponse**

- **status** must be *Accepted*

* Step 45:

Message: **ReportDERControlRequest**

- **requestId** matches requestId from GetDERControlRequest of step 43.

Field corresponding to configuring <CS supported controlType1>

- ***[0].id** must be *control1*

Field corresponding to configuring <CS supported controlType2>

- *Corresponding field should be omitted.*

Tool validations
<p>* Step 48: <i>Get scheduled controls</i></p> <p>Message: GetDERControlResponse</p> <ul style="list-style-type: none">- status must be <i>Accepted</i> <p>* Step 49:</p> <p>Message: ReportDERControlRequest</p> <ul style="list-style-type: none">- requestId matches requestId from GetDERControlRequest of step 47. <p>Field corresponding to configuring <CS supported controlType1></p> <ul style="list-style-type: none">- <i>Corresponding field should be omitted.</i> <p>Field corresponding to configuring <CS supported controlType2></p> <ul style="list-style-type: none">- *[0].id must be <i>control4</i>
<p>Post scenario validations:</p> <p>N/a</p>

TC_R_103_CS: Configure DER control - Validations

Test case name	Configure DER control settings at CS - validations
Test case Id	TC_R_103_CS
Use case Id(s)	R04
Requirement(s)	R04.FR.01, R04.FR.30, R04.FR.36, R04.FR.41, R04.FR.42, R04.FR.43
System under test	Charging Station
Description	Test System tries to set, clear or get DER controls which is not supported, set, or supported by the CS.
Purpose	To check if the CS returns the correct status.
Prerequisite(s)	EV supports DER Control

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 sends a SetDERControlResponse	1. The Test System sends a SetDERControlRequest with controlType <CS Unsupported controlType> controlId control1 priority 6
4. The Charging Station sends a GetDERControlResponse	3. The Test System sends a GetDERControlRequest with isDefault true controlType <CS Unsupported controlType>
6. The Charging Station sends a ClearDERControlResponse	5. The Test System sends a ClearDERControlRequest with isDefault false
8. The Charging Station sends a GetDERControlResponse	7. The Test System sends a GetDERControlRequest with isDefault false controlType <CS supported controlType1>
10. The Charging Station sends a ClearDERControlResponse	9. The Test System sends a ClearDERControlRequest with isDefault false controlType <CS supported controlType2>
12. The Charging Station sends a ClearDERControlResponse	11. The Test System sends a ClearDERControlRequest with isDefault false controlId 12345
14. The Charging Station sends a ClearDERControlResponse	13. The Test System sends a ClearDERControlRequest with isDefault false controlType <CS Unsupported controlType>

Tool validations
<div><div>* Step 2:</div><div>Message: SetDERControlResponse</div><div>- status must be <i>NotSupported</i></div><div>* Step 4:</div><div>Message: GetDERControlResponse</div><div>- status must be <i>NotSupported</i></div><div>* Step 6:</div><div>Message: ClearDERControlResponse</div><div>- status must be <i>Accepted</i></div><div>* Step 8:</div><div>Message: GetDERControlResponse</div><div>- status must be <i>NotFound</i></div><div>* Step 10:</div><div>Message: ClearDERControlResponse</div><div>- status must be <i>NotFound</i></div><div>* Step 12:</div><div>Message: ClearDERControlResponse</div><div>- status must be <i>NotFound</i></div><div>* Step 14:</div><div>Message: ClearDERControlResponse</div><div>- status must be <i>NotSupported</i></div></div>
<div>Post scenario validations:</div> <div>N/a</div>

TC_R_104_CS: Configure DER control - Superseding future DER control

Test case name	Configure DER control settings at CS - superseding future DER control
Test case Id	TC_R_104_CS
Use case Id(s)	R04
Requirement(s)	R04.FR.02, R04.FR.03, R04.FR.05, R04.FR.06
System under test	Charging Station
Description	Test System tries to set a DER control which is supported by the CS and no other controls are set on the CS.
Purpose	To check if the CS behaves correctly when setting controlTypes with different priorities.
Prerequisite(s)	EV supports DER Control

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 sends a ClearDERControlResponse	1. The Test System sends a ClearDERControlRequest with isDefault true
4. The Charging Station sends a ClearDERControlResponse	3. The Test System sends a ClearDERControlRequest with isDefault false
6. The Charging Station sends a SetDERControlResponse	5. The Test System sends a SetDERControlRequest with controlId control1 isDefault true controlType <CS supported controlType1> *.priority 5
8. The Charging Station sends a SetDERControlResponse	7. The Test System sends a SetDERControlRequest with controlId control2 isDefault true controlType <CS supported controlType1> *.priority 4
10. The Charging Station sends a GetDERControlResponse	9. The Test System sends a GetDERControlRequest with isDefault true
11. The Charging Station sends a ReportDERControlRequest	12. The Test System sends a ReportDERControlResponse
14. The Charging Station sends a SetDERControlResponse	13. The Test System sends a SetDERControlRequest with controlId control3 isDefault false controlType <CS supported controlType1> *.priority 5 *.startTime <Current DateTime + 2 hours>
16. The Charging Station sends a SetDERControlResponse	15. The Test System sends a SetDERControlRequest with controlId control4 isDefault false controlType <CS supported controlType1> *.priority 4 *.startTime <Current DateTime + 4 hours>
18. The Charging Station sends a GetDERControlResponse	17. The Test System sends a GetDERControlRequest with isDefault false

Main (Test scenario)	
19. The Charging Station sends a ReportDERControlRequest	20. The Test System sends a ReportDERControlResponse

Tool validations
<p>* Step 2: Message: ClearDERControlResponse - status must be <i>Accepted</i></p> <p>* Step 4: Message: ClearDERControlResponse - status must be <i>Accepted</i></p> <p>* Step 6: Message: SetDERControlResponse - status must be <i>Accepted</i></p> <p>* Step 8: Message: SetDERControlResponse - status must be <i>Accepted</i> - supersededIds[0] must be <i>control1</i></p> <p>* Step 10: Message: GetDERControlResponse - status must be <i>Accepted</i></p> <p>* Step 11: Message: ReportDERControlRequest Field corresponding to configuring <CS supported controlType1> - *[0].id must be <i>control1</i> - *[0].isDefault must be <i>true</i> - *[0].isSuperseded must be <i>true</i> - *[1].id must be <i>control2</i> - *[1].isDefault must be <i>true</i> - *[1].isSuperseded must be <i>false</i></p> <p>* Step 14: Message: SetDERControlResponse - status must be <i>Accepted</i></p> <p>* Step 16: Message: SetDERControlResponse - status must be <i>Accepted</i> - supersededIds[0] must be <i>control3</i></p> <p>* Step 18: Message: GetDERControlResponse - status must be <i>Accepted</i></p> <p>* Step 19: Message: ReportDERControlRequest Field corresponding to configuring <CS supported controlType1> - *[0].id must be <i>control3</i> - *[0].isDefault must be <i>false</i> - *[0].isSuperseded must be <i>true</i> - *[1].id must be <i>control4</i> - *[1].isDefault must be <i>false</i> - *[1].isSuperseded must be <i>false</i></p> <p>Post scenario validations: N/a</p>

TC_R_105_CS: Configure DER control - Superseding active DER control

Test case name	Configure DER control settings at CS - superseding active DER control
Test case Id	TC_R_105_CS
Use case Id(s)	R04
Requirement(s)	R04.FR.06, R04.FR.21, R04.FR.22
System under test	Charging Station
Description	Test System tries to set a DER control which is supported by the CS and no other controls are set on the CS.
Purpose	To check if the CS behaves correctly when setting controlTypes with different priorities.
Prerequisite(s)	EV supports DER Control

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 sends a ClearDERControlResponse	1. The Test System sends a ClearDERControlRequest with isDefault true
4. The Charging Station sends a ClearDERControlResponse	3. The Test System sends a ClearDERControlRequest with isDefault false
6. The Charging Station sends a SetDERControlResponse	5. The Test System sends a SetDERControlRequest with controlId control1 isDefault false controlType <CS supported controlType1> *.priority 6 *.startTime <Current DateTime> *.duration 300
7 The Charging Station sends a NotifyDERStartStopRequest	8. The Test System sends a NotifyDERStartStopResponse
10. The Charging Station sends a SetDERControlResponse	9. The Test System sends a SetDERControlRequest with controlId control2 isDefault false controlType <CS supported controlType1> *.priority 5 *.startTime <Current DateTime + 60 seconds> *.duration 60
<u>Note(s)</u> : After about 60 seconds...	
11 The Charging Station sends a NotifyDERStartStopRequest	12. The Test System sends a NotifyDERStartStopResponse
<u>Note(s)</u> : After about 60 seconds...	
13 The Charging Station sends a NotifyDERStartStopRequest	14. The Test System sends a NotifyDERStartStopResponse

Tool validations
<p>* Step 2:</p> <p>Message: ClearDERControlResponse</p> <ul style="list-style-type: none">- status must be <i>Accepted</i> <p>* Step 4:</p> <p>Message: ClearDERControlResponse</p> <ul style="list-style-type: none">- status must be <i>Accepted</i> <p>* Step 6:</p> <p>Message: SetDERControlResponse</p> <ul style="list-style-type: none">- status must be <i>Accepted</i>- supersededIds must be <i><omitted></i> <p>* Step 7:</p> <p>Message: NotifyDERStartStopRequest</p> <ul style="list-style-type: none">- controlId must be <i>control1</i>- started must be <i>true</i>- timestamp must be <i><Current DateTime></i>- supersededIds must be <i><omitted></i> <p>* Step 10:</p> <p>Message: SetDERControlResponse</p> <ul style="list-style-type: none">- status must be <i>Accepted</i>- supersededIds must be <i><omitted></i> <p>* Step 11:</p> <p>Message: NotifyDERStartStopRequest</p> <ul style="list-style-type: none">- controlId must be <i>control2</i>- started must be <i>true</i>- timestamp must be <i><Current DateTime></i>- supersededIds must be <i>control1</i> <p>* Step 13:</p> <p>Message: NotifyDERStartStopRequest</p> <ul style="list-style-type: none">- controlId must be <i>control2</i>- started must be <i>false</i>- timestamp must be <i><Current DateTime></i>- supersededIds must be <i>control1</i>
<p>Post scenario validations:</p> <p>N/a</p>

TC_R_106_CS: Configure DER control - Active DER control supersedes new DER control

Test case name	Configure DER control settings at CS - Active DER control supersedes new DER control
Test case Id	TC_R_106_CS
Use case Id(s)	R04
Requirement(s)	R04.FR.06, R04.FR.20, R04.FR.22
System under test	Charging Station
Description	Test System tries to set a DER control which is supported by the CS and no other controls are set on the CS.
Purpose	To check if the CS behaves correctly when setting controlTypes with different priorities.
Prerequisite(s)	EV supports DER Control

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 sends a ClearDERControlResponse	1. The Test System sends a ClearDERControlRequest with isDefault <i>true</i>
4. The Charging Station sends a ClearDERControlResponse	3. The Test System sends a ClearDERControlRequest with isDefault <i>false</i>
6. The Charging Station sends a SetDERControlResponse	5. The Test System sends a SetDERControlRequest with controlId <i>control1</i> isDefault <i>false</i> controlType <i><CS supported controlType1></i> *.priority <i>5</i> *.startTime <i><Current DateTime></i> *.duration <i>60</i>
7 The Charging Station sends a NotifyDERStartStopRequest	8. The Test System sends a NotifyDERStartStopResponse
10. The Charging Station sends a SetDERControlResponse	9. The Test System sends a SetDERControlRequest with controlId <i>control2</i> isDefault <i>false</i> controlType <i><CS supported controlType1></i> *.priority <i>6</i> *.startTime <i><Current DateTime + 20 seconds></i> *.duration <i>7200</i>
12. The Charging Station sends a GetDERControlResponse	11. The Test System sends a GetDERControlRequest with isDefault <i>false</i> controlType <i><CS supported controlType1></i>
13. The Charging Station sends a ReportDERControlRequest	14. The Test Systems sends a ReportDERControlResponse
Note: Wait ~60 seconds for control1 to end	
15. The Charging Station sends a NotifyDERStartStopRequest	16. The Test System sends a NotifyDERStartStopResponse

Tool validations
<p>* Step 2:</p> <p>Message: ClearDERControlResponse</p> <ul style="list-style-type: none"> - status must be <i>Accepted</i> <p>* Step 4:</p> <p>Message: ClearDERControlResponse</p> <ul style="list-style-type: none"> - status must be <i>Accepted</i> <p>* Step 6:</p> <p>Message: SetDERControlResponse</p> <ul style="list-style-type: none"> - status must be <i>Accepted</i> - supersededIds must be <i><omitted></i> <p>* Step 7:</p> <p>Message: NotifyDERStartStopRequest</p> <ul style="list-style-type: none"> - controlId must be <i>control1</i> - started must be <i>true</i> - timestamp must be <i><Current DateTime></i> - supersededIds must be <i><omitted></i> <p>* Step 10:</p> <p>Message: SetDERControlResponse</p> <ul style="list-style-type: none"> - status must be <i>Accepted</i> - supersededIds must be <i>control2</i> <p>* Step 12:</p> <p>Message: GetDERControlResponse</p> <ul style="list-style-type: none"> - status must be <i>Accepted</i> <p>* Step 13:</p> <p>Message ReportDERControlRequest</p> <p>Field corresponding to configuring <i><CS supported controlType1></i></p> <ul style="list-style-type: none"> - *[0].id must be <i>control1</i> - *[0].isDefault must be <i>false</i> - *[0].isSuperseded must be <i>false</i> - *[1].id must be <i>control2</i> - *[1].isDefault must be <i>false</i> - *[1].isSuperseded must be <i>true</i> <p>* Step 15:</p> <p>Message: NotifyDERStartStopRequest</p> <ul style="list-style-type: none"> - controlId must be <i>control1</i> - started must be <i>false</i>
<p>Post scenario validations:</p> <p>N/a</p>

TC_R_108_CS: Charging station reporting a DER event

Test case name	Charging station reporting a DER event
Test case Id	TC_R_108_CS
Use case Id(s)	R05
Requirement(s)	R05.FR.01, R05.FR.02, R05.FR.03, R05.FR.04
System under test	Charging Station
Description	Configure a DER Control which will always trigger and affect the charging rate of the running transaction, then add another DER Control to test overriding the previous one. After that, clearing all DER controls to test it reports that the alert is cleared. Any DER curve can be configured for this. Curves that are a function of power (WattPF, WattVar) or voltage (HV/LVMustTrip, VoltVar, VoltWatt) are usually easier to trigger than frequency-based curves.
Purpose	To check if the CS reports when DER controls are taking over.
Prerequisite(s)	CS supports DER Control (DCDERCtrlr or ACDERCtrlr is present for EVSE) CS is a DC charging station CS supports ISO15118-20

Before (Preparations)
Configuration State: - ISO15118Ctrlr.Enabled = true - V2XChargingCtrlr.V2XEnabled = true
Memory State: N/a
Reusable State(s): State is <i>EVConnectedPreSession</i> State is <i>Authorized15118</i> (for 15118-20) State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Note:</u> Wait 10 seconds after start of energy transfer before setting DER control	
2. The Charging Station sends a SetDERControlResponse	1. The Test System sends a SetDERControlRequest with controlId <i>control1</i> isDefault <i>true</i> controlType <i><Configured controlType1></i> curve <i><Configured curve></i> *.priority <i>6</i>
3. The Charging Station sends a NotifyDERAlarmRequest	4. The Test System responds with a NotifyDERAlarmResponse
6. The Charging Station sends a SetDERControlResponse	5. The Test System sends a SetDERControlRequest with controlId <i>control2</i> isDefault <i>true</i> controlType <i><Configured controlType2></i> curve <i><Configured curve2></i> *.priority <i>5</i>
7. The Charging Station sends a NotifyDERAlarmRequest	8. The Test System responds with a NotifyDERAlarmResponse
10. The Charging Station sends a ClearDERControlResponse	9. The Test System sends a ClearDERControlRequest with isDefault <i>true</i>
11. The Charging Station sends a NotifyDERAlarmRequest	12. The Test System responds with a NotifyDERAlarmResponse

Tool validations
<div><div>* Step 2:</div><div>Message: SetDERControlResponse</div><div><div>- status must be <i>Accepted</i></div></div></div> <div><div>* Step 3:</div><div>Message: NotifyDERAlarmRequest</div><div><div>- controlType must be <i>Configured controlType1</i></div><div>- gridEventFault must be <i><not omitted></i></div><div>- alarmEnded must be <i>false</i> or <i><absent></i></div><div>- timestamp must be <i><Current DateTime></i></div></div></div> <div><div>* Step 6:</div><div>Message: SetDERControlResponse</div><div><div>- status must be <i>Accepted</i></div></div></div> <div><div>* Step 7:</div><div>Message: NotifyDERAlarmRequest</div><div><div>- controlType must be <i>Configured controlType2</i></div><div>- gridEventFault must be <i><not omitted></i></div><div>- alarmEnded must be <i>false</i> or <i><absent></i></div><div>- timestamp must be <i><Current DateTime></i></div></div></div> <div><div>* Step 10:</div><div>Message: ClearDERControlResponse</div><div><div>- status must be <i>Accepted</i></div></div></div> <div><div>* Step 11:</div><div>Message: NotifyDERAlarmRequest</div><div><div>- controlType must be <i>Configured controlType2</i></div><div>- gridEventFault must be <i><not omitted></i></div><div>- alarmEnded must be <i>true</i></div><div>- timestamp must be <i><Current DateTime></i></div></div></div>
<div><div>Post scenario validations:</div><div>No other NotifyDERAlarmRequest shall be received, except those mentioned in the tool validation steps.</div></div>

S Battery Swapping

TC_S_102_CS: Battery Swap - Remote Start - not enough batteries

Test case name	Battery Swap - Remote Start - not enough batteries
Test case Id	TC_S_102_CS
Use case Id(s)	S02
Requirement(s)	S02.FR.04
System under test	Charging Station
Description	Charging station is able to support battery swapping.
Purpose	To verify whether the Charging Station is able to communicate when batteries are not available for swapping when remotely initiated by CSMS.
Prerequisite(s)	Charging Station supports battery swapping (BatterySwapCtrlr.Available is <i>true</i>) Not enough batteries are available for swapping.

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 RequestBatterySwapResponse	1. The Test System sends a RequestBatterySwapRequest with requestId <not omitted> idToken.idToken <Configured valid_idtoken_idtoken>

Tool validations
* Step 2: Message: RequestBatterySwapResponse - status must be <i>Rejected</i> - statusInfo.reasonCode must be <i>NoBatteryAvailable</i>
Post scenario validations: N/a

TC_S_104_CS: Battery Swap - Charging - Variables validation

Test case name	Battery Swap - Charging - Variables validation
Test case Id	TC_S_104_CS
Use case Id(s)	S04
Requirement(s)	S04.FR.08, S04.FR.09, S04.FR.10
System under test	Charging Station
Description	Charging station is able to support battery swapping.
Purpose	To verify whether the Charging Station is able to be configured correctly.
Prerequisite(s)	Charging Station supports battery swapping (BatterySwapCtrlr.Available is <i>true</i>)

Before (Preparations)
Configuration State: BatterySwapCtrlr.TargetSoc is 80 BatterySwapCtrlr.MaxSoc is 90
Memory State: N/a
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with GetBaseReportResponse	1. The Test System sends GetBaseReportRequest with requestId = <Generated requestId> reportBase = FullInventory
3. The Charging Station sends NotifyReportRequest	4. The Test System responds with NotifyReportResponse
5. The Charging Station sends NotifyReportRequest	6. The Test System responds with NotifyReportResponse
<u>Note:</u> If NotifyReportRequest.tbc is true in step 3 then steps 5 and 6 repeat until NotifyReportRequest.tbc is false in step 5.	
7. The Test System sends a SetVariablesRequest with setVariableData[0].component.name = BatterySwapCtrlr setVariableData[0].variable.name = TargetSoc setVariableData[0].attributeValue = "91"	8. The Charging Station responds with SetVariablesResponse
9. The Test System sends a SetVariablesRequest with setVariableData[0].component.name = BatterySwapCtrlr setVariableData[0].variable.name = MaxSoc setVariableData[0].attributeValue = "79"	10. The Charging Station responds with SetVariablesResponse
11. The Test System sends a SetVariablesRequest with setVariableData[0].component.name = BatterySwapCtrlr setVariableData[0].variable.name = TargetSoc setVariableData[0].attributeValue = "79"	12. The Charging Station responds with SetVariablesResponse
13. The Test System sends a SetVariablesRequest with setVariableData[0].component.name = BatterySwapCtrlr setVariableData[0].variable.name = MaxSoc setVariableData[0].attributeValue = "79"	14. The Charging Station responds with SetVariablesResponse

Tool validations

* Step 2:

Message: **GetBaseReportResponse**

- **status** must be *Accepted*
- **statusInfo** is absent or **statusInfo.reasonCode** = *"NoError"*

* Step 3:

Message: **NotifyReportRequest**

- **requestId** must be *<Generated requestId>*
- **generatedAt** must be *<timestamp at charging station>*
- **seqNo** must be *0*

* Step 5:

Message: **NotifyReportRequest**

- **requestId** must be *<Generated requestId>*
- **generatedAt** must be *<timestamp at charging station>*
- **seqNo** must be *<incremented seqNo>*

* Step 8:

Message: **SetVariablesResponse**

- **setVariableResult[0].attributeStatus** must be *Rejected*

* Step 10:

Message: **SetVariablesResponse**

- **setVariableResult[0].attributeStatus** must be *Rejected*

* Step 12:

Message: **SetVariablesResponse**

- **setVariableResult[0].attributeStatus** must be *Accepted*

* Step 14:

Message: **SetVariablesResponse**

- **setVariableResult[0].attributeStatus** must be *Accepted*

Post scenario validations:

In the collected data of all **NotifyReportRequest.reportData** it must contain:

- **baseReportData[0].component.name** must be *TxCtrlr*
- **baseReportData[0].component.variable.name** must be *TxStartPoint*
- **baseReportData[0].component.variableAttribute.type** is absent or must be *Actual*
- **baseReportData[0].component.variableAttribute.value** must be *EVConnected*
- **baseReportData[0].component.variableAttribute.mutability** must be *ReadOnly*
- **baseReportData[0].component.variableCharacteristics.dataType** must be *MemberList*
- **baseReportData[0].component.variableCharacteristics.valuesList** must be *EVConnected*
- **baseReportData[1].component.name** must be *TxCtrlr*
- **baseReportData[1].component.variable.name** must be *TxStopPoint*
- **baseReportData[1].component.variableAttribute.type** is absent or must be *Actual*
- **baseReportData[1].component.variableAttribute.value** must be *EVConnected*
- **baseReportData[1].component.variableAttribute.mutability** must be *ReadOnly*
- **baseReportData[1].component.variableCharacteristics.dataType** must be *MemberList*
- **baseReportData[1].component.variableCharacteristics.valuesList** must be *EVConnected*

TC_S_105_CS: Battery Swap - Charging - Battery Swap Charging

Test case name	Battery Swap - Charging - Battery Swap Charging
Test case Id	TC_S_105_CS
Use case Id(s)	S01, S04
Requirement(s)	S01.FR.01, S03.FR.01, S03.FR.02, S03.FR.03, S04.FR.01, S04.FR.02, S04.FR.07, S04.FR.06
System under test	Charging Station
Description	Charging station is able to support battery swapping.
Purpose	To verify whether the Charging Station communicates the status changes correctly to the CSMS during the charging process.
Prerequisite(s)	Charging Station supports battery swapping (BatterySwapCtrlr.Available is <i>true</i>) One battery with less or equal SoC than 96 percent

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. Test System sends GetVariablesRequest with: - variable.name = "SoC" - component.name = "BatteryCartridge" - component.evse.id = "<i>" - attributeType is omitted
<u>Note:</u> Step 1 and 2 request the SoC for each configured EVSE	
4. The Charging Station responds with GetVariablesResponse	3. Test System sends SetVariablesRequest with: - variable.name = "TargetSoc" - component.name = "BatterySwapCtrlr" - attributeValue = "<Highest returned SoC value from step 2, that is lower than 96 percent + 2>" - attributeType is omitted
6. The Charging Station responds with GetVariablesResponse	5. Test System sends SetVariablesRequest with: - variable.name = "MaxSoc" - component.name = "BatterySwapCtrlr" - attributeValue = "<Highest returned SoC value from step 2, that is lower than 96 percent + 4>" - attributeType is omitted
7. The Charging Station notifies the CSMS about the status change of the slot.	8. The Test System responds accordingly.
<u>Note:</u> Step 7 shall be triggered when the battery's SoC reaches TargetSoc	
9. The Charging Station sends a TransactionEventRequest	10. The Test System responds with a TransactionEventResponse
<u>Note:</u> Step 9 shall be triggered when the battery's SoC reaches the MaxSoc	

Tool validations
<div>* Step 7: Message: StatusNotificationRequest - connectorStatus must be <i>Available</i> Message: NotifyEventRequest - eventData[0].trigger must be <i>Delta</i> - eventData[0].actualValue must be <i>Available</i> - eventData[0].component.name must be <i>Connector</i> - eventData[0].variable.name must be <i>AvailabilityState</i> * Step 9: Message: TransactionEventRequest - eventType must be <i>Updated</i> - triggerReason must be <i>EnergyLimitReached</i> - transactionInfo.chargingState must be <i>SuspendedEVSE</i></div>
<div>Post scenario validations: N/a</div>

Memory states

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 (Test scenario)

Charging Station	CSMS
1. The Test System closes the WebSocket connection AND does not accept a reconnect.	
<u>Manual Action</u> : Drive EV into parking bay.	
<u>Manual Action</u> : Connect the EV and EVSE.	
<u>Manual Action</u> : Present idToken.	
<u>Manual Action</u> : Present the same idToken as used to start the transaction.	
<u>Manual Action</u> : Disconnect the EV and EVSE.	
<u>Manual Action</u> : Drive EV out of parking bay.	
2. The Test System accepts reconnection attempt from the Charging Station.	

Tool validations

N/a

Post scenario validations:

TransactionEventRequest messages are stored in the queue of the Charging Station.

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 (Test scenario)

Charging Station	CSMS
2. The Charging Station responds with a InstallCertificateResponse	1. The Test System sends a InstallCertificateRequest with certificateType is <Specified certificateType> certificate is <Corresponding certificate>

Tool validations

* Step 2:

Message: **InstallCertificateResponse**- **status** must be *Accepted*

Post scenario validations:

Certificate of the specified certificateType is stored at the Charging Station.

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: - AuthCacheCtrlr.Enabled is <i>true</i> (If implemented)
Memory State: N/a
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>ParkingBayoccupied</i>	
2. Execute Reusable State <i>Authorized</i>	
<u>Note(s)</u> : Step 3 and onwards are executed in case the idToken at step 2 was Accepted.	
3. Execute Reusable State <i>EVConnectedPreSession</i>	
4. Execute Reusable State <i>EnergyTransferStarted</i>	
5. Execute Reusable State <i>StopAuthorized</i>	
6. Execute Reusable State <i>EVDisconnected</i>	
7. Execute Reusable State <i>ParkingBayUnoccupied</i>	

Tool validations
N/a

IdTokenCached15118

State	IdTokenCached15118
System under test	Charging Station
Description	A 15118-idToken is stored in the Authorization Cache of the Charging Station.

Before (Preparations)
Configuration State: - AuthCacheCtrlr.Enabled is <i>true</i> (If implemented)
Memory State: N/a
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>ParkingBayoccupied</i>	
2. Execute Reusable State <i>EVConnectedPreSession</i>	
3. Execute Reusable State <i>Authorized15118</i>	
4. Execute Reusable State <i>EnergyTransferStarted</i>	
5. Execute Reusable State <i>StopAuthorized</i> (Remote)	
6. Execute Reusable State <i>EVDisconnected</i>	
7. Execute Reusable State <i>ParkingBayUnoccupied</i>	

Tool validations
N/a

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: LocalAuthListCtrlr.Enabled 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 SendLocalListResponse	1. The Test System 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 scenario validations: N/a

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 (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a SetChargingProfileResponse	1. The Test System sends a SetChargingProfileRequest with chargingProfile <Provided chargingProfile>

Tool validations
* Step 2: (Message: SetChargingProfileResponse) status is <i>Accepted</i>
Post scenario validations: N/a

SetDisplayMessage

State	SetDisplayMessage
System under test	Charging Station
Description	This will set a display message at 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 Charging Station responds with a SetDisplayMessageResponse	1. The Test System sends a SetDisplayMessageRequest with message.id <Generated displayMessageId> message.priority <Configured priority> message.state <Omitted, unless specifically described at the testcase> message.display <Configured display component variable> (Omitted when left empty) message.message.format <Configured Message Format> message.message.content <Configured Message>

Tool validations
* Step 2: (Message: SetDisplayMessageResponse) status is Accepted
Post scenario validations: N/a

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 Test System sends a TriggerMessageRequest With requestedMessage <i>SignChargingStationCertificate</i>
3 The Charging Station sends a SignCertificateRequest	4. The Test System responds with a SignCertificateResponse With status <i>Accepted</i>
6. The Charging Station responds with a CertificateSignedResponse	5. The Test System sends a CertificateSignedRequest With certificateChain <i><Certificate generated from the received CSR from step 3 and signed by the provided CSMS Root certificate></i> certificateType <i>ChargingStationCertificate</i>
<i>If the certificate is valid, then Charging Station should reconnect with the new certificate. Test System waits some time for a reconnection, and if that does not occur, will drop the connection to force a reconnection.</i>	
7. The Charging Station reconnects.	
8. If Charging Station rebooted: The Charging Station sends a BootNotificationRequest	9. Test System responds with a BootNotificationResponse .

Tool validations

* Step 2:

Message: **TriggerMessageResponse**- **status** must be *Accepted*

* Step 3:

Message: **SignCertificateRequest**- **csr** must contain *<An CSR that meets the following requirements:**When using RSA or DSA the key must be at least 2048 bits long.**and when using elliptic curve cryptography the key must be at least 224 bits long.**The received CSR must be transmitted as described in RFC 2986 and then encoded in Privacy-Enhanced Mail (PEM) format.>*

* Step 6:

Message: **CertificateSignedResponse**- **status** must be *Accepted*

* Step 7:

Charging Station must reconnect with new certificate.

Post scenario validations:

N/a

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 implemented

ISO15118Ctrlr.CountryName is *NL* if implemented

ISO15118Ctrlr.OrganizationName is configured vendorId if implemented

Test System 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 Test System sends a TriggerMessageRequest With requestedMessage <i>SignV2GCertificate</i> EVSE EVSE (having an <i>secCId</i>) returned in the <i>GetReportResponse</i> or omitted in case none is available
3 The Charging Station sends a SignCertificateRequest	4. The Test System responds with a SignCertificateResponse With status <i>Accepted</i>
6. The Charging Station responds with a CertificateSignedResponse	5. The Test System 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 <i>V2GCertificate</i>
<u>Note(s)</u> : Steps 1, 2, 3, 4, 5, and 6 are repeated for all returned <i>secCId</i> s	

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

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.

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 (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a ResetResponse	1. The Test System sends a ResetRequest with type Immediate

Tool validations
* Step 2: Message: ResetResponse - status must be <i>Accepted</i>
Post scenario validations: State is <i>Booting</i>

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):

If **Product subtype** is *Battery Swapping Station*, then execute [BootedBatterySwapping](#) instead

Main (Test 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 <i>Accepted</i>	1. The Test System sends a ResetRequest
3. The Charging Station sends a BootNotificationRequest	4. The Test System responds with a BootNotificationResponse with status <i>Accepted</i>
5. The Charging Station notifies the CSMS about the current state of all connectors.	6. The Test System responds accordingly.
7 The Charging Station sends a SecurityEventNotificationRequest	8 The Test System responds with a SecurityEventNotificationResponse

Tool validations

* Step 2:

Message: **ResetResponse**

- **status** *Accepted*

* Step 5:

Message: **StatusNotificationRequest**

- **connectorStatus** *Available*

- **evseld** 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 scenario validations:

State is *Booted*

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 (Test scenario)

Charging Station	CSMS
2. The Charging Station responds with a ReserveNowResponse	1. The Test System sends a ReserveNowRequest with evseld is <Specified evseld (Configured evseld 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. <u>Note(s):</u> - The Test System expects that the Charging Station sets the <i>availabilityState</i> of the EVSE and corresponding connectors to <i>Reserved</i> . - Reporting the <i>AvailabilityState</i> of the EVSE component itself is optional.	4. The Test System responds accordingly.

Tool validations

* Step 2:

Message: **ReserveNowResponse**- **status** must be *Accepted*

* Step 3:

Message: **StatusNotificationRequest**- **evseld** 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 scenario validations:

State is *Reserved*

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 (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a ChangeAvailabilityResponse	1. The Test System sends a ChangeAvailabilityRequest with operationalStatus <i>Inoperative</i> evse.id <Specified evseld> 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 Test System responds accordingly.

Tool validations
<p>* Step 2: Message ChangeAvailabilityResponse - status <i>Accepted</i></p> <p>* Step 3: Message: StatusNotificationRequest - connectorStatus <i>Unavailable</i> - evseld <Specified evseld> - connectorId <Specified connectorId></p> <p>Message: NotifyEventRequest - eventData[0].trigger <i>Delta</i> - eventData[0].actualValue <i>"Unavailable"</i> - eventData[0].component.name <i>"ChargingStation" / EVSE / Connector</i> - eventData[0].variable.name <i>"AvailabilityState"</i></p>
<p>Post scenario validations: State is <i>Reserved</i></p>

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 (Test scenario)	
Charging Station	CSMS
<u>Manual Action:</u> Drive EV into parking bay.	
<u>Note(s):</u> - This State is optional (Even when TxStartPoint contains ParkingBayOccupancy).	
1. The Charging Station sends a TransactionEventRequest <u>Note(s):</u> - This step needs to be executed when TxStartPoint contains ParkingBayOccupancy AND the EV entered the parking bay.	2. The Test System responds with a TransactionEventResponse

Tool validations
* Step 1: Message: TransactionEventRequest - triggerReason must be EVDetected
Post scenario validations: State is ParkingBayOccupied

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 **Product subtype** is *Battery Swapping Station*, then execute *EVConnectedPreSessionBatterySwapping* instead
 If **State** is NOT *ParkingBayOccupied* then execute **Reusable State** *ParkingBayOccupied*

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 Test System responds accordingly.
3. The Charging Station sends a TransactionEventRequest <u>Note(s):</u> - This step is executed with eventType Started if: TxStartPoint contains EVConnected OR TxStartPoint contains PowerPathClosed and the test case was initiated from the state <i>Authorized</i> OR - This step is executed with eventType Updated if: TxStartPoint contains ParkingBayOccupancy OR TxStartPoint contains Authorized and the test case was initiated from the state <i>Authorized</i> .	4. The Test System responds with a TransactionEventResponse

Tool validations

* Step 1:

Message: **StatusNotificationRequest**

- **evseld** <configured evseld>
- for the connector involved in the transaction: **connectorStatus** *Occupied*
- optionally for other connectors of the same EVSE: **connectorStatus** *Available* or *Unavailable*

Message: **NotifyEventRequest**

- **eventData[0].trigger** must be *Delta*
- **eventData[0].component.name** must be *Connector*
- **eventData[0].variable.name** must be *AvailabilityState*
- **evse.id** <configured evseld>
- for the connector involved in the transaction: **eventData[0].actualValue** *Occupied*
- optionally for other connectors of the same EVSE: **eventData[0].actualValue** *Available* or *Unavailable*

* 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 evseld>
- **connector.id** <configured connectorId>

Tool validations
Post scenario validations: State is <i>EVConnectedPreSession</i>

Authorized

State	Authorized
System under test	Charging Station
Description	<p>This state will prepare the Charging Station, so that the transaction is authorized. This can be done in two ways (The default way is configurable at Test System. This will be used when the calling testcase does not define which one to use.):</p> <p>A. Using local authorization</p> <p>B. Using a RequestStartTransactionRequest</p>

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): If Product subtype is <i>Battery Swapping Station</i> , then execute <i>AuthorizedBatterySwapping</i> instead If State is NOT <i>ParkingBayOccupied</i> OR <i>EVConnectedPreSession</i> , then execute Reusable State <i>ParkingBayOccupied</i>

Main A (Test scenario)	
Charging Station	CSMS
<u>Manual Action:</u> Present idToken.	
<p>1. The Charging Station sends an AuthorizeRequest</p> <p><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 Test System) OR the idToken is cached. In case the idToken is used for a reservation, sending the AuthorizeRequest message is optional.</p>	<p>2. The Test System responds with an AuthorizeResponse with idTokenInfo.status Accepted</p>
<p>3. The Charging Station sends a TransactionEventRequest</p> <p><u>Note(s):</u> - This step is executed with eventType Started if: TxStartPoint contains Authorized OR TxStartPoint contains PowerPathClosed and the test case was initiated from the state EVConnectedPreSession OR - This step is executed with eventType Updated if: TxStartPoint contains ParkingBayOccupancy OR TxStartPoint contains EVConnected and the test case was initiated from the state EVConnectedPreSession.</p>	<p>4. The Test System responds with a TransactionEventResponse</p> <p><u>Note(s):</u> - The first TransactionEventRequest sent after authorization contains the idToken field, unless a Start button was used to start the transaction. In case there is an idToken used, the TransactionEventResponse of this request message contains idTokenInfo with status Accepted</p>

Tool validations
<p>* Step 1:</p> <p>Message: AuthorizeRequest</p> <ul style="list-style-type: none"> - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type> <p>* Step 3:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - triggerReason must be <i>Authorized</i> - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type>

Main B (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a RequestStartTransactionResponse	1. The Test System sends a RequestStartTransactionRequest with idToken.idToken <Configured valid_idtoken_idtoken> idToken.type <Configured valid_idtoken_type> evseld <Configured evseld>
3. The Charging Station sends an AuthorizeRequest <u>Note(s):</u> - This step needs to be executed when AuthCtrlr.AuthorizeRemoteStart is true, unless (AuthEnabled is implemented with mutability <i>ReadOnly</i> AND the value is set to false) OR the idToken is cached. In case the idToken is used for a reservation, sending the AuthorizeRequest message is optional.	4. The Test System responds with an AuthorizeResponse with idTokenInfo.status <i>Accepted</i>
5. The Charging Station sends a TransactionEventRequest <u>Note(s):</u> - This step is executed with eventType <i>Started</i> if: TxStartPoint contains <i>Authorized</i> OR TxStartPoint contains <i>PowerPathClosed</i> and the test case was initiated from the state <i>EVConnectedPreSession</i> OR - This step is executed with eventType <i>Updated</i> if: TxStartPoint contains <i>ParkingBayOccupancy</i> OR TxStartPoint contains <i>EVConnected</i> and the test case was initiated from the state <i>EVConnectedPreSession</i> .	6. The Test System responds with a TransactionEventResponse <u>Note(s):</u> - The first TransactionEventRequest sent after authorization contains the idToken field. The TransactionEventResponse of this request message contains idTokenInfo with status <i>Accepted</i>

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 **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 scenario validations:

State is *Authorized*

Authorized15118

State	Authorized15118
System under test	Charging Station
Description	This state will prepare the Charging Station, so that the transaction is authorized, using plug and charge. Fields that are specific to ISO 15118-20 are marked with [15118-20]
Prerequisite	State is EVConnectedPreSession

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 an <code>AuthorizeRequest</code> <u>Note(s):</u> - The test case should be robust enough to also handle a <code>GetCertificateStatusRequest</code> or <code>Get15118EVCertificateRequest</code> and then expect the <code>AuthorizeRequest</code> .	2. The Test System responds with an <code>AuthorizeResponse</code> with <code>idTokenInfo.status</code> <i>Accepted</i> <code>certificateStatus</code> <i>Accepted</i> <code>allowedEnergyTransfer</code> [<i>AC_single_phase, AC_two_phase, AC_three_phase, DC, AC_BPT, AC_BPT_DER, AC_DER, DC_BPT, DC_ACDP, DC_ACDP_BPT, WPT</i>] [15118-20]
3. The Charging Station sends a <code>TransactionEventRequest</code> <u>Note(s):</u> - This step is executed with <code>eventType</code> <i>Started</i> if: <code>TxStartPoint</code> contains <i>Authorized</i> OR <code>TxStartPoint</code> contains <i>PowerPathClosed</i> (prerequisite state EVConnectedPreSession) OR - This step is executed with <code>eventType</code> <i>Updated</i> if: <code>TxStartPoint</code> contains <i>ParkingBayOccupancy</i> OR <code>TxStartPoint</code> contains <i>EVConnected</i> (prerequisite state EVConnectedPreSession)	4. The Test System responds with a <code>TransactionEventResponse</code> <u>Note(s):</u> - The first <code>TransactionEventRequest</code> sent after authorization contains the <code>idToken</code> field.
5. The Charging Station sends a <code>NotifyEVChargingNeedsRequest</code>	6. Test System responds with a <code>NotifyEVChargingNeedsResponse</code> with <code>status</code> <i>Accepted</i>

Tool validations

* Step 1:

Message: **AuthorizeRequest**

- **idToken.type** *eMAID*

- **iso15118CertificateHashData** *<Not omitted>*

If **ISO15118Ctrlr.CentralContractValidationAllowed** is *true*:

- **certificate** *<Not omitted>*

If **ISO15118Ctrlr.CentralContractValidationAllowed** is *false*:

- **certificate** *<Omitted>*

* Step 3:

Message: **TransactionEventRequest**

- **triggerReason** must be *Authorized*

- **idToken** *<Not omitted>*

- **transactionInfo.transactionId** must be *<transactionId>*

- **idToken.additionalInfo.additionalIdToken** must be *<EVCCID>* **[15118-20]**

- **idToken.additionalInfo.type** must be *EVCCID* **[15118-20]**

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 **Product subtype** is *Battery Swapping Station*, then execute [EnergyTransferStartedBatterySwapping](#) instead

If **State** is NOT [Authorized](#) then execute **Reusable State** [Authorized](#)

If connector <configured connectorId> state is not occupied, then execute **Reusable State** [EVConnectedPreSession](#)

Main (Test scenario)

Charging Station	CSMS
1. The Charging Station sends a TransactionEventRequest <u>Note(s):</u> - This step is only executed if TxStartPoint is <i>DataSigned</i> , in which case the eventType will be <i>Started</i> .	2. The Test System responds with a TransactionEventResponse
3. The Charging Station sends a TransactionEventRequest <u>Note(s):</u> - Step 3 and 4 are optional.	4. The Test System responds with a TransactionEventResponse
5. The Charging Station sends a TransactionEventRequest <u>Note(s):</u> - This step is executed with eventType <i>Started</i> if: TxStartPoint contains <i>EnergyTransfer</i> OR - This step is executed with eventType <i>Updated</i> if: TxStartPoint contains <i>ParkingBayOccupancy</i> OR TxStartPoint contains <i>Authorized</i> OR TxStartPoint contains <i>EVConnected</i> OR TxStartPoint contains <i>PowerPathClosed</i> OR TxStartPoint contains <i>DataSigned</i>	6. The Test System responds with a TransactionEventResponse

Tool validations

* Step 1:

Message: **TransactionEventRequest**

- **triggerReason** must be *SignedDataReceived*

* Step 3:

Message: **TransactionEventRequest**

- **triggerReason** must be *ChargingStateChanged*

- **transactionInfo.chargingState** must be *SuspendedEVSE*

* Step 5:

Message: **TransactionEventRequest**

- **triggerReason** must be *ChargingStateChanged*

- **transactionInfo.chargingState** must be *Charging*

Tool validations
Post scenario validations: State is <i>EnergyTransferStarted</i>

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 (Test 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	2. The Test System responds with a TransactionEventResponse

Tool validations
<p>* Step 1:</p> <p>Message: TransactionEventRequest</p> <p>IF TxStopPoint contains <i>EnergyTransfer</i> THEN:</p> <ul style="list-style-type: none"> - triggerReason must be <i>ChargingStateChanged</i> (If chargingState = <i>SuspendedEV</i>) - transactionInfo.chargingState must be <i>EVConnected</i> OR <i>SuspendedEV</i> - transactionInfo.stoppedReason must be <i>StoppedByEV</i> - eventType must be <i>Ended</i> <p>ELSE:</p> <ul style="list-style-type: none"> - triggerReason must be <i>ChargingStateChanged</i> - transactionInfo.chargingState must be <i>SuspendedEV</i> - eventType must be <i>Updated</i> <p>END</p>
<p>Post scenario validations:</p> <p>State is <i>EnergyTransferSuspended</i></p>

StopAuthorized

State	StopAuthorized
System under test	Charging Station
Description	<p>This state will prepare the Charging Station, so that it is in a state where the charging session is authorized to stop. This can be done in two ways (Configurable at Test System):</p> <p>A. Using local authorization</p> <p>B. Using a RequestStopTransactionRequest</p>

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): If State is NOT <i>EnergyTransferStarted</i> then execute Reusable State <i>EnergyTransferStarted</i>
<p>Note: The Test System will wait a number of seconds equal to the configured <i><TransactionDuration></i>, before proceeding to the Main stage.</p>

Main A (Test scenario)	
Charging Station	CSMS
<u>Notes(s)</u> : The tool will wait for <Configured Transaction Duration> seconds	
<u>Manual Action</u> : Present the same idToken as used to start the transaction.	
1. The Charging Station sends a TransactionEventRequest	2. The Test System responds with a TransactionEventResponse With idTokenInfo.status is <i>Accepted</i>
<u>Note(s)</u> : This step is optional	
3. The Charging Station sends a TransactionEventRequest	4. The Test System responds with a TransactionEventResponse With idTokenInfo.status is <i>Accepted</i>

Tool validations
<p>* Step 1:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - triggerReason must be <i>StopAuthorized</i> - idToken omit OR - idToken.idToken <i><Configured valid_idtoken_idtoken></i> AND - idToken.type <i><Configured valid_idtoken_type></i> <p>* Step 3:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - triggerReason must be <i>ChargingStateChanged</i> - transactionInfo.chargingState must be <i>EVConnected</i> <p>If TxStopPoint contains <i>Authorized</i> or <i>PowerPathClosed</i> or <i>EnergyTransfer</i></p> <p>Then the last of the two TransactionEventRequest messages from step 1 and 3 needs to contain:</p> <ul style="list-style-type: none"> - eventType must be <i>Ended</i> - transactionInfo.stoppedReason must be <i>Local</i> or omitted <p>Else</p> <p>Then both TransactionEventRequest messages need to contain:</p> <ul style="list-style-type: none"> - eventType must be <i>Updated</i>

Main B (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a RequestStopTransactionResponse	1. The Test System sends a RequestStopTransactionRequest with transactionId <transactionId provided by the Charging Station in TransactionEventRequest >
3. The Charging Station sends a TransactionEventRequest	4. The Test System responds with a TransactionEventResponse
<u>Note(s)</u> : This step is optional	
5. The Charging Station sends a TransactionEventRequest	6. The Test System responds with a TransactionEventResponse With idTokenInfo.status is <i>Accepted</i>

Tool validations
<p>* Step 2: Message: RequestStopTransactionResponse - status must be <i>Accepted</i></p> <p>* Step 3: Message: TransactionEventRequest - triggerReason must be <i>RemoteStop</i></p> <p>* Step 5: Message: TransactionEventRequest - triggerReason must be <i>ChargingStateChanged</i> - transactionInfo.chargingState must be <i>EVConnected</i></p> <p>If TxStopPoint contains Authorized or PowerPathClosed or EnergyTransfer Then the last of the two TransactionEventRequest messages from step 3 and 5 needs to contain: - eventType must be <i>Ended</i> - transactionInfo.stoppedReason must be <i>Remote</i></p> <p>Else Then both TransactionEventRequest messages need to contain: - eventType must be <i>Updated</i></p>
<p>Post scenario validations: State is <i>StopAuthorized</i></p>

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 (Test scenario)

Charging Station	CSMS
1. The Charging Station sends a TransactionEventRequest <u>Note(s):</u> - This step needs to be executed when the transaction has NOT been ended already. So in the case TxStopPoint contains Authorized OR PowerPathClosed	2. The Test System responds with a TransactionEventResponse
3. The Charging Station sends a TransactionEventRequest <u>Note(s):</u> - 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 Test System 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 scenario validations:

State is *EVConnectedPostSession*

Deauthorized

State	Deauthorized
System under test	Charging Station
Description	This reusable state will set the Charging Station to a state in which the transaction will be deauthorized.
Prerequisite	Reusable State <i>Authorized</i> is executed.

Before (Preparations)

Configuration State: N/a
Memory State: N/a
Reusable State(s): Continues executing transaction reusable states in order until the first TransactionEventRequest is received based on the configured TxStartPoint.

Main (Test scenario)

Charging Station	CSMS
1. The Charging Station sends a TransactionEventRequest	2. The Test System responds with a TransactionEventResponse With idTokenInfo.status is <i>Invalid</i>
3. The Charging Station sends a TransactionEventRequest <u>Note(s):</u> - Step 3 and 4 are relevant when AuthCtrlr.StopTxOnInvalidId is <i>true</i>	4. The Test System responds with a TransactionEventResponse
6. The Charging Station sends a UnlockConnectorResponse	5. The Test System sends a UnlockConnectorRequest <u>Note(s):</u> - Step 5 and 6 are executed when the connector is locked and the transaction gets deauthorized.

Tool validations

* Step 1:
 Message: **TransactionEventRequest**
 - **idToken.idToken** <Configured valid_idtoken_idtoken> AND
 - **idToken.type** <Configured valid_idtoken_type>

* Step 3:
 Message: **TransactionEventRequest**
 - **triggerReason** must be *Deauthorized*
 - **transactionInfo.chargingState** must be *EVConnected* or *SuspendedEVSE*
 If TxStopPoint contains *Authorized* or *PowerPathClosed* or *EnergyTransfer*
 Then :
 - **eventType** must be *Ended*
 - **stoppedReason** must be *DeAuthorized*
 Else:
 - **eventType** must be *Updated*

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 **Product subtype** is *Battery Swapping Station*, then execute [EVDisconnectedBatterySwapping](#) instead
 If **State** is NOT [EVConnectedPostSession](#) then execute **Reusable State** [EVConnectedPostSession](#)

Main (Test scenario)

Charging Station	CSMS
<u>Manual Action:</u> <i>Disconnect the EV and EVSE.</i>	
<u>Note:</u> Steps 3 & 4 are allowed to occur before step 1.	
1. The Charging Station notifies the CSMS about the status change of the connector.	2. The Test System responds accordingly.
3. The Charging Station sends a TransactionEventRequest <u>Note(s):</u> - <i>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</i>	4. The Test System responds with a TransactionEventResponse

Tool validations

* Step 1:

Message: **StatusNotificationRequest**

- **connectorStatus** must be *Available*
- **evseld** must be *<configured evseld>*
- **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 evseld>*
- **eventData[0].component.evse.connectorId** must be *<configured connectorId>*

* Step 3:

Message: **TransactionEventRequest**

- **triggerReason** must be *EVCommunicationLost*
- **transactionInfo.chargingState** must be *Idle*

Post scenario validations:

State is *EVDisconnected*

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 Product subtype is <i>Battery Swapping Station</i> THEN execute Battery Swapping reusable states instead. <i>Note: ParkingBayOccupancy is not applicable for a Battery Swapping station, however this reusable state is used as an end state to run through the full transaction flow, therefore it needs refer to the end state applicable for the Battery Swapping station instead.</i> <i>Battery Swapping:</i> IF BatterySwapCtrlr.SwapOrder is <i>In-Out</i> or not implemented AND State is NOT <i>EVDisconnectedBatterySwapping</i> THEN execute Reusable State <i>EVDisconnectedBatterySwapping</i> ELSE IF BatterySwapCtrlr.SwapOrder is <i>Out-In</i> AND State is NOT <i>EnergyTransferStartedBatterySwapping</i> THEN execute Reusable State <i>EnergyTransferStartedBatterySwapping</i> END <i>Standard:</i> If State is NOT <i>EVDisconnected</i> then execute Reusable State <i>EVDisconnected</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action:</u> Drive EV out of parking bay.	
1. The Charging Station sends a TransactionEventRequest <u>Note(s):</u> - This step needs to be executed when TxStopPoint contains <i>ParkingBayOccupancy</i> AND the transaction has NOT been ended already. So in the case TxStopPoint contains Authorized OR EnergyTransfer OR PowerPathClosed OR DataSigned OR EVConnected.	2. The Test System responds with a TransactionEventResponse

Tool validations
* Step 1: Message: TransactionEventRequest - triggerReason must be <i>EVDeparted</i> - If the Test System is configured to stop transactions using a <i>RequestStopTransactionRequest</i> message then transactionInfo.stoppedReason must be <i>Remote</i> Else transactionInfo.stoppedReason must be <i>Local</i> - eventType must be <i>Ended</i>
Post scenario validations: State is <i>ParkingBayUnoccupied</i>

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 (Test scenario)	
Charging Station	CSMS
1. The Test System closes the WebSocket connection AND does not accept a reconnect.	
<u>Manual Action:</u> Drive EV into parking bay.	
<u>Manual Action:</u> Present idToken.	
<u>Manual Action:</u> Connect the EV and EVSE.	
2. The Test System accepts reconnection attempt from the Charging Station.	

Tool validations
N/a
Post scenario validations: N/a

RenegotiateChargingLimits

State	RenegotiateChargingLimits
System under test	Charging Station
Description	A version for the ISO15118 can be specified as argument to this reusable state. This version is exposed as variable <i><iso15118Version></i> and can either be <i>iso15118-2</i> or <i>iso15118-20</i> . If not specified <i>iso15118-2</i> will be assumed.
Prerequisite	

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> Renegotiate EV Charging Limits	
1. The Charging Station sends a NotifyEVChargingNeedsRequest with evseld <i><Configured evseld></i>	2. The Test System responds with a NotifyEVChargingNeedsResponse with status <i>Accepted</i>
4. The Charging Station responds with a SetChargingProfileResponse with status <i>Accepted</i>	3. The Test System sends a SetChargingProfileRequest with chargingProfile : .chargingProfilePurpose <i>TxProfile</i> .transactionId <i><Provided transactionId from before></i> chargingProfile.chargingSchedule[0]: .duration <i>300</i> .chargingRateUnit <i><Configured chargingRateUnit></i> <i>Note: If <Configured chargingRateUnit> is W, then the limit field will be multiplied by 1000.</i> .chargingSchedulePeriod[0].startPeriod <i>0</i> If <i><Configured chargingRateUnit></i> is W: .chargingSchedulePeriod[0].limit <i>10000</i> else: .chargingSchedulePeriod[0].limit <i>10</i>
5. The Charging Station sends a NotifyEVChargingScheduleRequest with evseld <i><Configured evseld></i>	6. The Test System responds with a NotifyEVChargingScheduleResponse with status <i>Accepted</i>
<u>Note:</u> Steps 5 and 6 are optional. The Charging Station will only send a NotifyEVChargingScheduleRequest when the EV returns a charging profile.	

Tool validations
<div><div>* Step 1:</div><div>Message: NotifyEVChargingNeedsRequest)</div><div><div>- evseld <Configured evseld></div><div>- if (<iso15118Version> is iso15118-2 or omitted) and chargingNeeds.requestedEnergyTransfer is DC:</div><div>- chargingNeeds.acChargingParameters must be <omitted></div><div>- chargingNeeds.dcChargingParameters must be <not omitted></div><div>- chargingNeeds.v2xChargingParameters must be <not omitted></div><div>- end if</div><div>- if (<iso15118Version> is iso15118-2 or omitted) and chargingNeeds.requestedEnergyTransfer is AC:</div><div>- chargingNeeds.acChargingParameters must be <not omitted></div><div>- chargingNeeds.dcChargingParameters must be <omitted></div><div>- chargingNeeds.v2xChargingParameters must be <not omitted></div><div>- end if</div><div>- if <iso15118Version> is iso15118-20:</div><div>- chargingNeeds.acChargingParameters must be <omitted></div><div>- chargingNeeds.dcChargingParameters must be <omitted></div><div>- chargingNeeds.v2xChargingParameters must be <not omitted></div><div>- end if</div><div>- chargingSchedule must be <not omitted></div></div><div><div>* Step 4:</div><div>Message: SetChargingProfileResponse)</div><div><div>- status Accepted</div></div></div><div><div>* Step 5:</div><div>Message: NotifyEVChargingScheduleRequest)</div><div><div>- evseld <Configured evseld></div></div></div></div>
<div><div>Post scenario validations:</div><div>N/a</div></div>

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 (Test scenario)

Charging Station	CSMS
2. The Charging Station responds with a GetInstalledCertificateIdsResponse	1. The Test System 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 scenario validations:

Certificate of the specified certificateType is retrieved from the Charging Station.

RebootBeforeFirmwareInstallation

State	RebootBeforeFirmwareInstallation
System under test	Charging Station
Description	The Charging Station needs to reboot before firmware <u>installation</u> .

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 <i>InstallRebooting</i>	2. The Test System responds with a FirmwareStatusNotificationResponse
<u>Note:</u> 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 Test System responds with a BootNotificationResponse with status <i>Accepted</i>
5. The Charging Station notifies the CSMS about the current state of all connectors.	6. The Test System responds accordingly.
7. The Charging Station sends a FirmwareStatusNotificationRequest With status <i>Installing</i>	8. The Test System responds with a FirmwareStatusNotificationResponse

Tool validations
<p>* Step 1: Message FirmwareStatusNotificationRequest - status <i>InstallRebooting</i></p> <p>* Step 3: Message BootNotificationRequest - reason <i>FirmwareUpdate</i></p> <p>* Step 7: Message FirmwareStatusNotificationRequest - status <i>Installing</i></p>
Post scenario validations: N/a

RebootBeforeFirmwareActivation

State	RebootBeforeFirmwareActivation
System under test	Charging Station
Description	The Charging Station needs to reboot before firmware <u>activation</u> .

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 <code>FirmwareStatusNotificationRequest</code> With status <i>InstallRebooting</i> <u>Note(s):</u> - This step is optional. However it is recommended to notify the CSMS before rebooting the Charging Station to activate the new firmware.	2. The Test System responds with a <code>FirmwareStatusNotificationResponse</code>
3. The Charging Station sends a <code>BootNotificationRequest</code>	4. The Test System responds with a <code>BootNotificationResponse</code> with status <i>Accepted</i>
5. The Charging Station notifies the CSMS about the current state of all connectors.	6. The Test System responds accordingly.

Tool validations
* Step 1: Message <code>FirmwareStatusNotificationRequest</code> - status <i>InstallRebooting</i> * Step 3: Message <code>BootNotificationRequest</code> - reason <i>FirmwareUpdate</i>
Post scenario validations: N/a

BaseReportCommunicated

State	BaseReportCommunicated
System under test	Charging Station
Description	Retrieves the FullInventory base report from a Charging Station and makes it available for usage in test cases.
Exposes	<baseReportData>

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 GetBaseReportResponse	1. The Test System sends GetBaseReportRequest with requestId = <Generated requestId> reportBase = FullInventory
3. The Charging Station sends NotifyReportRequest	4. The Test System responds with NotifyReportResponse
Note(s): - If NotifyReportRequest.tbc is True in Step 3 then step 3 and 4 will be repeated	
All NotifyReportRequest.reportData are exposed as <baseReportData>	

Tool validations
* Step 2: Message GetBaseReportResponse - status Accepted
Post scenario validations: N/a

MonitoringReportCommunicated

State	MonitoringReportCommunicated
System under test	Charging Station
Description	<p>Retrieves the full monitoring report from a Charging Station and makes it available for usage in test cases.</p> <p>It is possible to specify reset to a specific MonitoringBase before retrieving the MonitoringReport using the argument.</p> <p>Argument is exposed variable <code><resetToMonitoringBase></code> and has valid values:</p> <ul style="list-style-type: none"> - <i>All</i> - set monitoring base to <i>All</i> - <i>FactoryDefault</i> - set monitoring base to <i>FactoryDefault</i> - <i>HardWiredOnly</i> - set monitoring base to <i>HardWiredOnly</i> - <i>NoReset</i> - do not set monitoring base <p>If argument is not specified <i>NoReset</i> will be assumed.</p>
Exposes	<code><monitoringReportData></code>

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 SetMonitoringBaseResponse <u>Note:</u> Skip if <code><resetToMonitoringBase></code> is <i>NoReset</i>	1. The Test System sends a SetMonitoringBaseRequest with monitoringBase <code><resetToMonitoringBase></code> <u>Note:</u> Skip if <code><resetToMonitoringBase></code> is <i>NoReset</i>
4. The Charging Station responds with GetMonitoringReportResponse	3. The Test System sends GetMonitoringReportRequest with requestId = <code><Generated requestId></code>
5. The Charging Station sends NotifyMonitoringReportRequest	6. The Test System responds with NotifyMonitoringReportResponse
<u>Note(s):</u> - If NotifyMonitoringReportRequest.tbc is <i>True</i> in Step 5 then step 5 and 6 will be repeated	
All NotifyMonitoringReportRequest.monitor are exposed as <code><monitoringReportData></code>	

Tool validations
* Step 2: Message SetMonitoringBaseResponse - status <i>Accepted</i> * Step 4: Message GetMonitoringReportResponse - status <i>Accepted</i>
Post scenario validations: N/a

Iso1511820V2xControlBpt

State	Iso1511820V2xControlBpt
System under test	Charging Station
Description	Configures the EV to use <i>DynamicControl</i> or <i>ScheduledControl</i> and <i>AC_BPT</i> or <i>DC_BPT</i> energy transfer mode. Whether to use <i>DynamicControl</i> or <i>ScheduledControl</i> is defined by the testcase. <i>DynamicControl</i> is assumed as a default when nothing is specified.
Exposes	<transactionId>

Before (Preparations)
Configuration State: - ISO15118Ctrlr.Enabled is true - V2XChargingCtrlr.V2XEnabled is true
Memory State: N/a
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
Agree on energy transfer mode	
<u>Manual Action:</u> Use EV to communicate EV charging limits:	
For AC charging station:	
requestedEnergyTransfer AC_BPT	
availableEnergyTransfer [AC_single_phase, AC_two_phase, AC_three_phase, AC_BPT, AC_BPT_DER, AC_DER]	
For DC charging station:	
requestedEnergyTransfer DC_BPT	
availableEnergyTransfer [DC, DC_BPT, DC_ACDP, DC_ACDP_BPT]	
controlMode DynamicControl or ScheduledControl	
maxChargePower 10000 (W)	
maxDischargePower 5000W (W)	

Tool validations
N/a
Post scenario validations: N/a

DefaultTariffConfigured

State	DefaultTariffConfigured
System under test	Charging Station
Description	Configures the Charging Station with a default tariff for all EVSE's.
Exposes	<defaultTariff>

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

Main (Test scenario)	
Charging Station	CSMS
1. The Test System sends a SetDefaultTariffRequest with evseld 0 tariff.tariffId <i>Test System1</i> tariff.currency <i>EUR</i> tariff.energy.taxRate[0].type <i>MyTax</i> tariff.energy.taxRate[0].tax 21 tariff.energy.prices[0].priceKwh 0.50	2. The Charging Station responds with SetDefaultTariffResponse status = <i>Accepted</i>
SetDefaultTariffRequest.tariff of step 1 is exposed as <defaultTariff>	

Tool validations
* Step 2: Message SetDefaultTariffResponse - status must be Accepted
Post scenario validations: N/a

BootedBatterySwapping

State	BootedBatterySwapping
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.

Before (Preparations)

Configuration State:

N/a

Memory State:

N/a

Reusable State(s):

N/a

Main (Test scenario)

Charging Station	CSMS
Manual Action: <i>Power cycle the Charging Station.</i> OR execute step 1 and 2, depending on the testcase.	
2. The Charging Station responds with a ResetResponse with status Accepted	1. The Test System sends a ResetRequest
3. The Charging Station sends a BootNotificationRequest	4. The Test System responds with a BootNotificationResponse with status Accepted
5. The Charging Station notifies the CSMS about the current state of all connectors.	6. The Test System responds accordingly.
7 The Charging Station sends a SecurityEventNotificationRequest	8 The Test System responds with a SecurityEventNotificationResponse

Tool validations

* Step 2:

Message: **ResetResponse**- **status Accepted**

* Step 5:

Message: **StatusNotificationRequest**- **connectorStatus Available** OR *Occupied*- **evseld** not 0- **connectorId** not 0Message: **NotifyEventRequest**- **eventData[0].trigger Delta**- **eventData[0].actualValue "Available"** OR *"Occupied"*- **eventData[0].component.name "Connector"**- **eventData[0].variable.name "AvailabilityState"**

* Step 7:

Message: **SecurityEventNotificationRequest**- **type** must be *StartupOfTheDevice* OR *ResetOrReboot*

Post scenario validations:

State is *Booted*

AuthorizedBatterySwapping

State	AuthorizedBatterySwapping
System under test	Charging Station

State	AuthorizedBatterySwapping
Description	This state will prepare the Charging Station, so that the EV driver is authorized to perform a battery swap. This can be done in two ways (The default way is configurable at Test System. This will be used when the calling testcase does not define which one to use.): A. Using local authorization B. Using a RequestBatterySwapRequest
Prerequisite	All battery slots need to be empty.

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

Main A (Test scenario)	
Charging Station	CSMS
<u>Manual Action:</u> Present <i>idToken</i> .	
1. The Charging Station sends an <i>AuthorizeRequest</i> <u>Note(s):</u> - This step needs to be executed, unless (<i>AuthEnabled</i> is implemented with mutability <i>ReadOnly</i> AND the value is set to <i>false</i>) OR a start button as described at Use case C02 is used (This must be configured at the Test System) OR the <i>idToken</i> is cached. In case the <i>idToken</i> is used for a reservation, sending the <i>AuthorizeRequest</i> message is optional.	2. The Test System responds with an <i>AuthorizeResponse</i> with <i>idTokenInfo.status</i> <i>Accepted</i>

Tool validations
* Step 1: Message: <i>AuthorizeRequest</i> - <i>idToken.idToken</i> <Configured <i>valid_idtoken_idtoken</i> > - <i>idToken.type</i> <Configured <i>valid_idtoken_type</i> >

Main B (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a <i>RequestBatterySwapResponse</i>	1. The Test System sends a <i>RequestBatterySwapRequest</i> with <i>requestId</i> <Configured <i>requestId</i>> <i>idToken.idToken</i> <Configured <i>valid_idtoken_idtoken</i>>

Tool validations
* Step 2: Message: <i>RequestBatterySwapResponse</i> - <i>status</i> must be <i>Accepted</i>
Post scenario validations: State is <i>AuthorizedBatterySwapping</i>

EVConnectedPreSessionBatterySwapping

State	EVConnectedPreSessionBatterySwapping
System under test	Charging Station
Description	This state will prepare the Charging Station, so that one or more batteries have been inserted into the slots.

Before (Preparations)

Configuration State:

N/a

Memory State:

N/a

Reusable State(s):

IF **BatterySwapCtrlr.SwapOrder** is *In-Out* or not implemented AND **State** is NOT [AuthorizedBatterySwapping](#)
 THEN execute **Reusable State** [AuthorizedBatterySwapping](#)
 ELSE IF **BatterySwapCtrlr.SwapOrder** is *Out-In* AND **State** is NOT [EVDisconnectedBatterySwapping](#)
 THEN execute **Reusable State** [EVDisconnectedBatterySwapping](#)
 END

Main (Test scenario)

Charging Station	CSMS
<u>Manual Action:</u> <i>Insert one or more batteries in slots</i>	
1. The Charging Station notifies the CSMS about the status change of the slot.	2. The Test System responds accordingly.
3. The Charging Station sends a TransactionEventRequest	4. The Test System responds with a TransactionEventResponse
<u>Note:</u> <i>Steps 1 through 4 are repeated for each battery inserted</i>	
5. The Charging Station sends a BatterySwapRequest	6. The Test System responds with a BatterySwapResponse

Tool validations
<p>* Step 1:</p> <p>Message: StatusNotificationRequest</p> <ul style="list-style-type: none"> - connectorStatus must be <i>Occupied</i> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none"> - eventData[0].trigger must be <i>Delta</i> - eventData[0].actualValue must be <i>Occupied</i> - eventData[0].component.name must be <i>Connector</i> - eventData[0].variable.name must be <i>AvailabilityState</i> <p>* Step 3:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - eventType must be <i>Started</i> - triggerReason must be <i>CablePluggedIn</i> - evse must be provided - evse.connectorId must be provided - transactionInfo.chargingState may be <i>Charging</i> (May be reported in a separate <i>TransactionEventRequest</i>) - idToken.type <i>Central</i> OR <i>NoAuthorization</i> <p>IF idToken.type <i>Central</i></p> <ul style="list-style-type: none"> - idToken.idToken <Configured <i>BatterySwap_Idtoken</i>> <p>ELSE IF idToken.type <i>NoAuthorization</i></p> <ul style="list-style-type: none"> - idToken.idToken <Empty string> <p>END</p> <p>* Step 5:</p> <p>Message: BatterySwapRequest</p> <ul style="list-style-type: none"> - eventType must be <i>BatteryIn</i> - requestId must be <Configured <i>requestId</i>> - idToken.idToken must be <Configured <i>valid_idtoken_idtoken</i>> <p>For each inserted battery:</p> <ul style="list-style-type: none"> - batteryData.evseId must be <not omitted> - batteryData.serialNumber must be <not omitted> - batteryData.soC must be <not omitted> - batteryData.soH must be <not omitted>
<p>Post scenario validations:</p> <p>State is <i>EVConnectedPreSessionBatterySwapping</i></p>

EnergyTransferStartedBatterySwapping

State	EnergyTransferStartedBatterySwapping
System under test	Charging Station
Description	This state will prepare the Charging Station, so that one or more batteries that have been inserted are charging.

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): If State is NOT EVConnectedPreSessionBatterySwapping , then execute Reusable State EVConnectedPreSessionBatterySwapping

Main (Test scenario)	
Charging Station	CSMS
1. The Charging Station sends a TransactionEventRequest	2. The Test System responds with a TransactionEventResponse
<u>Notes:</u> - Steps 1 and 2 are applicable if the Charging Station did not already report chargingState Charging at reusable state EVConnectedPreSessionBatterySwapping - Steps 1 and 2 are repeated for each battery inserted	

Tool validations
* Step 5: Message: TransactionEventRequest - eventType must be <i>Updated</i> - triggerReason must be <i>ChargingStateChanged</i> - transactionInfo.chargingState must be <i>Charging</i>
Post scenario validations: State is <i>EnergyTransferStartedBatterySwapping</i>

EVDIsconnectedBatterySwapping

State	EVDIsconnectedBatterySwapping
System under test	Charging Station
Description	This state will prepare the Charging Station, so that one or more batteries have been taken out from the slots.

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): IF BatterySwapCtrlr.SwapOrder is <i>In-Out</i> or not implemented AND State is NOT <i>EnergyTransferStartedBatterySwapping</i> THEN execute Reusable State <i>EnergyTransferStartedBatterySwapping</i> ELSE IF BatterySwapCtrlr.SwapOrder is <i>Out-In</i> AND State is NOT <i>AuthorizedBatterySwapping</i> THEN execute Reusable State <i>AuthorizedBatterySwapping</i> END

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action:</u> Take out one or more batteries from slots	
1. The Charging Station notifies the CSMS about the status change of the slot.	2. The Test System responds accordingly.
3. The Charging Station sends a TransactionEventRequest	4. The Test System responds with a TransactionEventResponse
<u>Note:</u> Steps 1 through 4 are repeated for each battery taken out	
5. The Charging Station sends a BatterySwapRequest	6. The Test System responds with a BatterySwapResponse

Tool validations
<p>* Step 1:</p> <p>Message: StatusNotificationRequest</p> <ul style="list-style-type: none">- connectorStatus must be <i>Available</i> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none">- eventData[0].trigger must be <i>Delta</i>- eventData[0].actualValue must be <i>Available</i>- eventData[0].component.name must be <i>Connector</i>- eventData[0].variable.name must be <i>AvailabilityState</i> <p>* Step 3:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none">- eventType must be <i>Ended</i>- triggerReason must be <i>EVCommunicationLost</i>- transactionInfo.chargingState must be <i>Idle</i>- transactionInfo.stoppedReason must be <i>EVDisconnected</i>- idToken.type <i>Central</i> OR <i>NoAuthorization</i> <p>IF idToken.type <i>Central</i></p> <ul style="list-style-type: none">- idToken.idToken <<i>Configured BatterySwap_Idtoken</i>> <p>ELSE IF idToken.type <i>NoAuthorization</i></p> <ul style="list-style-type: none">- idToken.idToken <<i>Empty string</i>> <p>END</p> <p>* Step 5:</p> <p>Message: BatterySwapRequest</p> <ul style="list-style-type: none">- eventType must be <i>BatteryOut</i>- requestId must be <<i>Configured requestId</i>>- idToken.idToken must be <<i>Configured valid_idtoken_idtoken</i>> <p>For each inserted battery:</p> <ul style="list-style-type: none">- batteryData.evseId must be <<i>not omitted</i>>- batteryData.serialNumber must be <<i>not omitted</i>>- batteryData.soC must be <<i>not omitted</i>>- batteryData.soH must be <<i>not omitted</i>>
<p>Post scenario validations:</p> <p>State is <i>EVDisconnectedBatterySwapping</i></p>

AuthorizedPaymentTerminal

State	AuthorizedPaymentTerminal
System under test	Charging Station
Description	<p>This state will prepare the Charging Station, so that the transaction is authorized via a payment terminal. This can be done in two ways (The default way is configurable at Test System. This will be used when the calling testcase does not define which one to use.):</p> <p>A. Using local authorization (Integrated payment terminal)</p> <p>B. Using a RequestStartTransactionRequest (Payment kiosk with communication channel to the CSMS)</p>

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): If State is NOT <i>ParkingBayOccupied</i> OR <i>EVConnectedPreSession</i> , then execute Reusable State <i>ParkingBayOccupied</i>

Main A (Test scenario)	
Charging Station	CSMS
<u>Manual action:</u> Present a payment card to the payment terminal.	
1. The Charging Station sends an AuthorizeRequest <u>Note(s):</u> - This step needs to be executed, unless PaymentCtrlr.AuthorizeDirectPayment = false	2. The Test System responds with an AuthorizeResponse with idTokenInfo.status Accepted
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 or (EVConnected, in the case this testcase was initiated from state EVConnectedPreSession.)	4. The Test System responds with a TransactionEventResponse <u>Note(s):</u> - 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
<p>* Step 1:</p> <p>Message AuthorizeRequest</p> <ul style="list-style-type: none"> - idToken must be <Not empty> - idToken.type must be <i>DirectPayment</i> - idToken.additionalInfo[0].additionalIdToken must be <not omitted> - idToken.additionalInfo[0].type must be <CardLast4Digits> <p>* Step 3:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - triggerReason must be <i>Authorized</i> - idToken must be <Not empty> - idToken.type must be <i>DirectPayment</i> - idToken.additionalInfo[0].additionalIdToken must be <not omitted> - idToken.additionalInfo[0].type must be <i>CardLast4Digits</i>

Main B (Test scenario)	
Charging Station	CSMS
2. The Charging Station responds with a RequestStartTransactionResponse	1. The Test System sends a RequestStartTransactionRequest with evseld <Configured evseld> idToken.idToken <Configured pspref_idtoken> idToken.type <i>DirectPayment</i>
3. The Charging Station sends an AuthorizeRequest <u>Note(s):</u> - This step needs to be executed when AuthCtrlr.AuthorizeRemoteStart is true, unless PaymentCtrlr.AuthorizeDirectPayment = false	4. The Test System responds with an AuthorizeResponse with idTokenInfo.status <i>Accepted</i>
5. The Charging Station sends a TransactionEventRequest <u>Note(s):</u> - This step needs to be executed when TxStartPoint contains <i>Authorized</i> OR the transaction already started. So in the case TxStartPoint contains <i>ParkingBayOccupancy</i> or (<i>EVConnected</i> , in the case this testcase was initiated from state <i>EVConnectedPreSession</i> .)	6. The Test System responds with a TransactionEventResponse <u>Note(s):</u> - The first <i>TransactionEventRequest</i> sent after authorization contains the idToken field. The <i>TransactionEventResponse</i> of this request message contains idTokenInfo with status <i>Accepted</i>

Tool validations
<p>* Step 2:</p> <p>Message: RequestStartTransactionResponse</p> <ul style="list-style-type: none"> - status must be <i>Accepted</i> <p>If the transaction has already been started, so if TxStartPoint contains <i>ParkingBayOccupancy</i> OR (<Configured TxStartPoint> contains <i>EVConnected</i> AND State pre reusable state execution was <i>EVConnectedPreSession</i>) then</p> <ul style="list-style-type: none"> - transactionId must be <Provided transactionId in first <i>TransactionEventRequest</i>> <p>* Step 3:</p> <p>Message AuthorizeRequest</p> <ul style="list-style-type: none"> - idToken must be <Not empty> - idToken.type must be <i>DirectPayment</i> <p>* Step 5:</p> <p>Message: TransactionEventRequest</p> <ul style="list-style-type: none"> - eventType Started if TxStartPoint is <i>Authorized</i> or <i>PowerPathClosed</i> and State is <i>EVConnectedPreSession</i>, else updated - triggerReason must be <i>RemoteStart</i> - transactionInfo.remoteStartId must be present. - idToken must be <Not empty> - idToken.type must be <i>DirectPayment</i> <p>Post scenario validations: State is <i>Authorized</i></p>

3. Test Cases Charging Station Management System

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.

A Security

TC_A_01_CSMS: Basic Authentication - Valid username/password combination

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 Test System.
Memory State: N/a
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
1. The Test System sends a HTTP upgrade request with an Authorization header, containing a username/password combination. <u>Note(s):</u> - 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 Test System sends a BootNotificationRequest	4. The CSMS responds with a BootNotificationResponse
5. The Test System notifies the CSMS about the current state of all connectors.	6. The CSMS responds accordingly.

Tool validations
* Step 4: Message: BootNotificationResponse - status must be <i>Accepted</i>
Post scenario validations: N/a

TC_A_02_CSMS: Basic Authentication - Username does not equal ChargingStationId

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
<p>1. The Test System sends a HTTP upgrade request with an Authorization header, containing a username/password combination.</p> <p><u>Note(s):</u> - The Authorization header is formatted as follows: AUTHORIZATION: Basic <Base64 encoded(<Configured ChargingStationId> + Invalid:<Configured basicAuthPassword>)></p>	<p>2. The CSMS validates the username/password combination AND rejects the connection upgrade request.</p>

Tool validations
N/a
Post scenario validations: N/a

TC_A_03_CSMS: Basic Authentication - Invalid password

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 Test System sends a HTTP upgrade request without an Authorization header.	2. The CSMS rejects the connection upgrade request.
3. The Test System sends a HTTP upgrade request with an Authorization header, containing a username/password combination. <u>Note(s):</u> - The Authorization header is formatted as follows: <i>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)>)></i>	4. The CSMS validates the username/password combination AND rejects the connection upgrade request.

Tool validations
N/a
Post scenario validations: N/a

TC_A_04_CSMS: TLS - server-side certificate - Valid certificate

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

Main (Test scenario)	
Charging Station	CSMS
1. The Test System 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 Test System performs the following actions: Send client certificate Client Key Exchange Certificate verify Change Cipher Spec Finished <u>Note(s):</u> - The client certificate is only sent when the CSMS uses security profile 3.	4. The CSMS performs the following actions: Change Cipher Spec Finished
5. The Test System sends a HTTP upgrade request to the CSMS <u>Note(s):</u> - The HTTP request only contains a username/password combination when the CSMS uses security profile 2.	6. The CSMS upgrades the connection to a (secured) WebSocket connection.
7. The Test System sends a BootNotificationRequest with reason <i>PowerUp</i> chargingStation.model <Configured model> chargingStation.vendorName <Configured vendorName>	8. The CSMS responds with a BootNotificationResponse

Main (Test scenario)	
<p>9. The Test System notifies the CSMS about the current state of all connectors.</p> <p>Message: StatusNotificationRequest</p> <ul style="list-style-type: none"> - connectorStatus <i>Available</i> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none"> - trigger <i>Delta</i> - actualValue <i>"Available"</i> - component.name <i>"Connector"</i> - variable.name <i>"AvailabilityState"</i> 	<p>10. The CSMS responds accordingly.</p>
Tool validations	
<p>* Step 3:</p> <p>The Test System validates the following before finishing the TLS handshake:</p> <ul style="list-style-type: none"> - The CSMS must use TLS version 1.2 or above <p>At least the following set of cipher suites must be supported:</p> <p>TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256</p> <p>AND</p> <p>TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384</p> <p>AND</p> <p>TLS_RSA_WITH_AES_128_GCM_SHA256</p> <p>AND</p> <p>TLS_RSA_WITH_AES_256_GCM_SHA384</p> <ul style="list-style-type: none"> - When using RSA or DSA the key must be at least 2048 bits long. <p>and when using elliptic curve cryptography the key must be at least 224 bits long.</p> <ul style="list-style-type: none"> - The received server side certificate must be transmitted in the X.509 format encoded in Privacy-Enhanced Mail (PEM) format. - The certificate must include a serial number. - The subject field of the certificate must contain a commonName RDN which consists of the FQDN of the endpoint of the server. <p><i>NOTE: If one of the above validations fails, the Test System can still proceed with the next steps of the testcase (if it is able to), but the testcase will FAIL and the Test System reports why it failed.</i></p> <p>* Step 8:</p> <p>Message: BootNotificationResponse</p> <p>with status <i>Accepted</i></p>	
<p>Post scenario validations:</p> <p>N/a</p>	

TC_A_06_CSMS: TLS - server-side certificate - TLS version too low

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 Test System 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 Test System 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 Test System performs the following actions: Send client certificate Client Key Exchange Certificate verify Change Cipher Spec Finished <u>Note(s):</u> - The client certificate is only sent when the CSMS uses security profile 3.	6. The CSMS performs the following actions: Change Cipher Spec Finished
7. The Test System sends a HTTP upgrade request to the CSMS <u>Note(s):</u> - The HTTP request only contains a username/password combination when the CSMS uses security profile 2.	8. The CSMS upgrades the connection to a (secured) WebSocket connection.
9. The Test System sends a BootNotificationRequest with reason <i>PowerUp</i> chargingStation.model <Configured model> chargingStation.vendorName <Configured vendorName>	10. The CSMS responds with a BootNotificationResponse

Main (Test scenario)	
<p>11. The Test System notifies the CSMS about the current state of all connectors.</p> <p>Message: StatusNotificationRequest</p> <ul style="list-style-type: none">- connectorStatus <i>Available</i> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none">- trigger <i>Delta</i>- actualValue <i>"Available"</i>- component.name <i>"Connector"</i>- variable.name <i>"AvailabilityState"</i>	<p>12. The CSMS responds accordingly.</p>

Tool validations
<p>* Step 10:</p> <p>Message: BootNotificationResponse</p> <ul style="list-style-type: none">- status <i>Accepted</i>
<p>Post scenario validations:</p> <p>N/a</p>

TC_A_07_CSMS: TLS - Client-side certificate - valid certificate

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 Test System 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 Test System 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 Test System sends a HTTP upgrade request to the CSMS	6. The CSMS upgrades the connection to a (secured) WebSocket connection.
7. The Test System sends a BootNotificationRequest with reason <i>PowerUp</i> chargingStation.model <Configured model> chargingStation.vendorName <Configured vendorName>	8. The CSMS responds with a BootNotificationResponse
9. The Test System notifies the CSMS about the current state of all connectors. Message: StatusNotificationRequest - connectorStatus <i>Available</i> Message: NotifyEventRequest - trigger <i>Delta</i> - actualValue <i>"Available"</i> - component.name <i>"Connector"</i> - variable.name <i>"AvailabilityState"</i>	10. The CSMS responds accordingly.

Tool validations
<p>* Step 3:</p> <p>The Test System validates the following before finishing the TLS handshake:</p> <ul style="list-style-type: none">- The CSMS must use TLS version 1.2 or above <p>At least the following set of cipher suites must be supported:</p> <p>TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256</p> <p>AND</p> <p>TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384</p> <p>AND</p> <p>TLS_RSA_WITH_AES_128_GCM_SHA256</p> <p>AND</p> <p>TLS_RSA_WITH_AES_256_GCM_SHA384</p> <p>* Step 8:</p> <p>Message: BootNotificationResponse</p> <p>with status <i>Accepted</i></p>
<p>Post scenario validations:</p> <p>N/a</p>

TC_A_08_CSMS: TLS - Client-side certificate - Invalid certificate

Test case name	TLS - Client-side certificate - Invalid certificate
Test case Id	TC_A_08_CSMS
Use case Id(s)	A00
Requirement(s)	A00.FR.405,A00.FR.407,A00.FR.409,A00.FR.410
System under test	CSMS
Description	The Charging Station uses a client-side certificate to identify itself to the CSMS, when using security profile 3.
Purpose	To verify whether the CSMS is able to terminate the connection when the received client certificate is invalid.
Prerequisite(s)	<ul style="list-style-type: none"> - The CSMS supports security profile 3 - This testcase can be executed multiple times, using different kinds of invalid certificates: Unknown certificate expired certificate certificate with commonName that does not equal the serial number of the Charging Station.

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

Main (Test scenario)	
Charging Station	CSMS
1. The Test System 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 Test System 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 Test System 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 Test System 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 Test System sends a HTTP upgrade request to the CSMS	10. The CSMS upgrades the connection to a (secured) WebSocket connection.
11. The Test System sends a BootNotificationRequest with reason <i>PowerUp</i> chargingStation.model <Configured model> chargingStation.vendorName <Configured vendorName>	12. The CSMS responds with a BootNotificationResponse

Main (Test scenario)	
<p>13. The Test System notifies the CSMS about the current state of all connectors.</p> <p>Message: StatusNotificationRequest</p> <ul style="list-style-type: none">- connectorStatus <i>Available</i> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none">- trigger <i>Delta</i>- actualValue <i>"Available"</i>- component.name <i>"Connector"</i>- variable.name <i>"AvailabilityState"</i>	<p>14. The CSMS responds accordingly.</p>

Tool validations
<p>* Step 12:</p> <p>Message: BootNotificationResponse with status <i>Accepted</i></p>
<p>Post scenario validations: N/a</p>

TC_A_09_CSMS: Update Charging Station Password for HTTP Basic Authentication - Accepted

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Test System 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 Test System sends a HTTP upgrade request with an Authorization header, containing a username/password combination (with the new <i>BasicAuthPassword</i>). <u>Note(s)</u> : - The Authorization header is formatted as follows: <i>AUTHORIZATION: Basic <Base64 encoded(<Configured ChargingStationId>:<NEW BasicAuthPassword>)></i>	4. The CSMS validates the username/password combination AND upgrades the connection to a (secured) WebSocket connection.
5. The Test System sends a BootNotificationRequest	6. The CSMS responds with a BootNotificationResponse
7. The Test System 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 <i>Accepted</i>
Post scenario validations: N/a

TC_A_10_CSMS: Update Charging Station Password for HTTP Basic Authentication - Rejected

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Test System responds with a SetVariablesResponse with status <i>Rejected</i>	<p>1. The CSMS sends a SetVariablesRequest with:</p> <p>setVariableData[1]:</p> <ul style="list-style-type: none"> - variable.name = "BasicAuthPassword" - component.name = "SecurityCtrlr" - attributeValue = "<NewPassword>"
	<p>3. The Test System sends a HTTP upgrade request with an Authorization header, containing a username/password combination (with the old <i>BasicAuthPassword</i>).</p> <p><u>Note(s):</u></p> <ul style="list-style-type: none"> - 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 Test System sends a BootNotificationRequest	6. The CSMS responds with a BootNotificationResponse
	7. The Test System notifies the CSMS about the current state of all connectors.
8. The CSMS responds accordingly.	

Tool validations
<p>* Step 1:</p> <p>Message: SetVariableRequest</p> <ul style="list-style-type: none">- variable.name = <i>"BasicAuthPassword"</i>- component.name = <i>"SecurityCtrlr"</i> <p>* Step 6:</p> <p>Message: BootNotificationResponse</p> <ul style="list-style-type: none">- status must be <i>Accepted</i>
<p>Post scenario validations:</p> <p>N/a</p>

TC_A_11_CSMS: Update Charging Station Certificate by request of CSMS - Success - Charging Station Certificate

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	
2. The Test System disconnects its current connection and reconnects to the CSMS with the new certificate.	3. The CSMS accepts the incoming connection request using the new certificate.

Tool validations
N/a
Post scenario validations: The Test System and the CSMS are connected.

TC_A_12_CSMS: Update Charging Station Certificate by request of CSMS - Success - V2G Certificate

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 Test System responds with a TriggerMessageResponse With status <i>Accepted</i>	1. The CSMS sends a TriggerMessageRequest
3 The Test System sends a SignCertificateRequest With csr <i>Generated CSR based on:</i> - <Configured Country> - <Configured Organization> - <Configured OrganizationalUnit> certificateType <i>V2GCertificate</i>	4. The CSMS responds with a SignCertificateResponse
6. The Test System responds with a CertificateSignedResponse With status <i>Accepted</i>	5. The CSMS sends a CertificateSignedRequest

Tool validations
<p>* Step 1:</p> <p>Message: TriggerMessageRequest</p> <ul style="list-style-type: none">- requestedMessage <i>SignV2GCertificate</i> <p>* Step 4:</p> <p>Message: SignCertificateResponse</p> <ul style="list-style-type: none">- status <i>Accepted</i> <p>* Step 5:</p> <p>Message: CertificateSignedRequest</p> <ul style="list-style-type: none">- certificateChain <i><Certificate generated from the received CSR from step 3 and signed by the V2G Root or SubCA certificate from the configured V2G certificate chain></i> <p><i>NOTE: The Test System 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.</i></p> <ul style="list-style-type: none">- 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.
<p>Post scenario validations:</p> <p>N/a</p>

TC_A_14_CSMS: Update Charging Station Certificate by request of CSMS - Invalid certificate

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 <code>TriggerMessageRequest</code> 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 Test System responds with a TriggerMessageResponse With status <i>Accepted</i>	1. The CSMS sends a TriggerMessageRequest
3 The Test System sends a SignCertificateRequest With csr <i><Configured CSR></i> certificateType <i>ChargingStationCertificate</i>	4. The CSMS responds with a SignCertificateResponse
6. The Test System responds with a CertificateSignedResponse With status <i>Rejected</i>	5. The CSMS sends a CertificateSignedRequest
7. The Test System sends a SecurityEventNotificationRequest with type = <i>InvalidChargingStationCertificate</i>	8. The CSMS responds with a SecurityEventNotificationResponse

Tool validations
* Step 1: Message: TriggerMessageRequest - requestedMessage <i>SignChargingStationCertificate</i> * Step 4: Message: SignCertificateResponse - status <i>Accepted</i>
Post scenario validations: N/a

TC_A_19_CSMS: Upgrade Charging Station Security Profile - Accepted

Test case name	Upgrade Charging Station Security Profile - Accepted
Test case Id	TC_A_19_CSMS
Use case Id(s)	A05
Requirement(s)	A05.FR.04, A05.FR.07
System under test	CSMS
Description	The CSMS updates the connection details on the Charging Station, to increase the security profile level.
Purpose	To verify if the CSMS is able to set a new network connection profile at one of the by the Charging Station defined configuration slots with a higher security profile than currently configured.
Prerequisite(s)	<ul style="list-style-type: none"> - Security profile must be set to 1 or 2. - If Security profile is set to 1, then a trusted certificate must be installed.

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

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action:</u> Request the CSMS to set a new NetworkConnectionProfile with a security profile level one higher than currently configured	
2. The Test System responds with a SetNetworkProfileResponse With status Accepted	1. The CSMS sends a SetNetworkProfileRequest
<u>Manual Action:</u> Request the CSMS to change the NetworkConfigurationPriority to one that contains the configurationSlot of the new NetworkConnectionProfile from step 1	
4. The Test System responds with a SetVariablesResponse with status Accepted	3. The CSMS sends a SetVariablesRequest
<u>Manual Action:</u> Request the CSMS to reboot the Charging Station	
6. The Test System responds with a ResetResponse with status Accepted	5. The CSMS sends a ResetRequest
7. The Test System reconnects to the CSMS using the new NetworkProfile, containing the upgraded security profile <Configured securityProfile + 1>.	8. The CSMS accepts the connection attempt.
9. Execute Reusable State Booted	
10. The Test System reconnects to the CSMS using the original NetworkProfile, containing the lower security profile.	11. The CSMS shall not accept the connection attempt.
<u>Note(s):</u> - This is done to ensure that the CSMS does not accept a connection using the lower security profile anymore.	

Main (Test scenario)	
<p>12. The Test System reconnects to the CSMS using the new NetworkProfile, containing the upgraded security profile <Configured securityProfile + 1>.</p> <p><u>Note(s):</u> - This is done to restore the connection before ending the testcase.</p>	<p>13. The CSMS accepts the connection attempt.</p>

Tool validations
<p>* Step 1: Message SetNetworkProfileRequest - connectionData.messageTimeout <Configured messageTimeout> - connectionData.ocppInterface <Configured ocppInterface> - connectionData.ocppTransport JSON - connectionData.ocppVersion OCPP20 - connectionData.securityProfile <Configured securityProfile + 1></p> <p>* Step 3: Message SetVariablesRequest setVariableData: - variable.name = "NetworkConfigurationPriority" - component.name = "OCPPCommCtrlr" - attributeValue = <contains configurationSlot provided at step 1></p> <p>* Step 11: When upgrading a Charging Station to a higher security profile, a CSMS has several options regarding which endpoint to use. This affects the way the CSMS is able to detect it needs to reject the incoming connection attempt.</p> <p>In case of having upgraded from security profile 2 to 3, but there is an incoming connection attempt using security profile 2: When the same endpoint is used, then it depends on the CSMS endpoint configuration. - When the CSMS does a full switch and only allows TLS handshakes when a client certificate is provided, then the TLS handshake is rejected. - When the CSMS only requires this Charging Station to use a client certificate, then it accepts the TLS handshake (because it will be unable to detect which Charging Station is connecting) and it rejects the HTTP request to establish the WebSocket connection.</p> <p>When a different port or a whole different endpoint is used for the upgrade, then on the original endpoint the CSMS accepts the TLS handshake and it rejects the HTTP request to establish the WebSocket connection (because this Charging Station is not allowed to connect with security profile 2 anymore).</p> <p>In case of security profile 1, the case is always the same. The CSMS shall always reject the HTTP request to establish the WebSocket connection, because TLS is required for this Charging Station.</p> <p>Post scenario validations: The Test System and the CSMS are connected.</p>

B Provisioning**TC_B_01_CSMS: Cold Boot Charging Station - Accepted**

Test case name	Cold Boot Charging Station - Accepted
Test case Id	TC_B_01_CSMS
Use case Id(s)	B01
Requirement(s)	B01.FR.02
System under test	CSMS
Description	The booting mechanism allows a Charging Station to provide some general information about the Charging Station to the CSMS on startup AND it allows the Charging Station to request whether it is allowed to start sending other OCPP messages.
Purpose	To verify whether the CSMS is able to accept the communications of a registered Charging Station.
Prerequisite(s)	N/a

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

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>Booted</i>	

Tool validations
N/a
Post scenario validations: N/a

TC_B_02_CSMS: Cold Boot Charging Station - Pending

Test case name	Cold Boot Charging Station - Pending
Test case Id	TC_B_02_CSMS
Use case Id(s)	B02
Requirement(s)	B02.FR.01, B02.FR.06
System under test	CSMS
Description	The booting mechanism allows a Charging Station to provide some general information about the Charging Station to the CSMS on startup AND it allows the Charging Station to request whether it is allowed to start sending other OCPP messages. The CSMS may respond to the BootNotificationRequest with status <i>Pending</i> . The <i>Pending</i> status can indicate that the CSMS wants to retrieve or set certain information on the Charging Station before it will accept the Charging Station.
Purpose	To verify whether the CSMS is able to accept the communications of a registered Charging Station.
Prerequisite(s)	The CSMS is configured to first respond to a BootNotificationRequest with status Pending .

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

Main (Test scenario)	
Charging Station	CSMS
1. The Test System sends a BootNotificationRequest with reason <i>PowerUp</i> chargingStation.model <Configured model> chargingStation.vendorName <Configured vendorName>	2. The CSMS responds with a BootNotificationResponse
<u>Note(s):</u> - If the interval in the BootNotificationResponse equals 0, the Test System will wait <Configured heartbeatInterval> seconds, before sending another BootNotificationRequest. - If the interval in the BootNotificationResponse > 0, the Test System 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 Test System will respond to these messages.	
3. The Test System sends a BootNotificationRequest with reason <i>PowerUp</i> chargingStation.model <Configured model> chargingStation.vendorName <Configured vendorName>	4. The CSMS responds with a BootNotificationResponse
5. The Test System notifies the CSMS about the current state of all connectors. Message: StatusNotificationRequest with connectorStatus <i>Available</i> Message: NotifyEventRequest with trigger <i>Delta</i> actualValue "Available" component.name "Connector" variable.name "AvailabilityState"	6. The CSMS responds accordingly.

Tool validations
<p>* Step 2:</p> <p>Message: BootNotificationResponse</p> <p>- status <i>Pending</i></p> <p>* Step 3:</p> <p>Message: BootNotificationResponse</p> <p>- status <i>Accepted</i></p>
<p>Post scenario validations:</p> <p>N/a</p>

TC_B_30_CSMS: Cold Boot Charging Station - Pending/Rejected - SecurityError

Test case name	Cold Boot Charging Station - Pending/Rejected - SecurityError
Test case Id	TC_B_30_CSMS
Use case Id(s)	B02/B03
Requirement(s)	B02.FR.09, B03.FR.07
System under test	CSMS
Description	The booting mechanism allows a Charging Station to provide some general information about the Charging Station to the CSMS on startup AND it allows the Charging Station to request whether it is allowed to start sending other OCPP messages. The CSMS may respond to the BootNotificationRequest with status <i>Pending</i> or <i>Rejected</i> . During this state, the Charging Station is not allowed to send RPC Framework: CALL message that is NOT a BootNotificationRequest or in case of status <i>Pending</i> , a message triggered by one of the following messages: TriggerMessageRequest, GetBaseReportRequest, GetReportRequest.
Purpose	To verify whether the CSMS is able to handle unauthorized messages from the Charging Station by responding with a SecurityError.
Prerequisite(s)	The CSMS is configured to first respond to a BootNotificationRequest with status <i>Pending</i> or <i>Rejected</i> .

Before (Preparations)

Configuration State:

N/a

Memory State:

N/a

Reusable State(s):

N/a

Main (Test scenario)

Charging Station	CSMS
1. The Test System sends a BootNotificationRequest with reason <i>PowerUp</i> chargingStation.model <Configured model> chargingStation.vendorName <Configured vendorName>	2. The CSMS responds with a BootNotificationResponse
3. The Test System notifies the CSMS about the current state of all connectors. Message: StatusNotificationRequest with connectorStatus <i>Available</i> Message: NotifyEventRequest with trigger <i>Delta</i> 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

TC_B_31_CSMS: Cold Boot Charging Station - Pending/Rejected - TriggerMessage

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 <i>Pending</i> or <i>Rejected</i> .

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

Main (Test scenario)	
Charging Station	CSMS
1. The Test System sends a BootNotificationRequest with reason <i>PowerUp</i> chargingStation.model <Configured model> chargingStation.vendorName <Configured vendorName>	2. The CSMS responds with a BootNotificationResponse
4. The Test System responds with a TriggerMessageResponse with status <i>Accepted</i>	3. The CSMS sends a TriggerMessageRequest
5. The Test System sends a BootNotificationRequest with reason <i>Triggered</i> chargingStation.model <Configured model> chargingStation.vendorName <Configured vendorName>	6. The CSMS responds with a BootNotificationResponse
7. The Test System notifies the CSMS about the current state of all connectors. Message: StatusNotificationRequest with connectorStatus <i>Available</i> Message: NotifyEventRequest with trigger <i>Delta</i> actualValue "Available" component.name "Connector" variable.name "AvailabilityState"	8. The CSMS responds accordingly.

Tool validations
<div>* Step 2: Message: BootNotificationResponse - status <i>Pending OR Rejected</i></div> <div>* Step 3: Message: TriggerMessageRequest - requestedMessage <i>BootNotification</i></div> <div>* Step 6: Message: BootNotificationResponse - status <i>Accepted</i></div>
<div>Post scenario validations: N/a</div>

TC_B_06_CSMS: Get Variables - single value

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. Test System responds with: GetVariablesResponse	1. <i>Manually request CSMS to get data for:</i> - OCPPCommCtrlr.OfflineThreshold

Tool validations
<p>* Step 1:</p> <p>Message: GetVariablesRequest with (in arbitrary order)</p> <p>getVariableData[0]:</p> <ul style="list-style-type: none"> - attributeType is at least absent or attributeType = <i>Actual</i>, but <i>Target</i>, <i>MinSet</i>, and <i>MaxSet</i> are also allowed - variable.name = "OfflineThreshold" - component.name = "OCPPCommCtrlr"
<p>Post scenario validations:</p> <p>Manually validate that CSMS has correctly read the requested variables.</p>

TC_B_07_CSMS: Get Variables - multiple values

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. Test System responds with: GetVariablesResponse	1. Manually request CSMS to get data for: - OCPPCommCtrlr.OfflineThreshold - AuthCtrlr.AuthorizeRemoteStart

Tool validations
<p>* Step 1:</p> <p>Message: GetVariablesRequest with (in arbitrary order)</p> <p>getVariableData[0]:</p> <ul style="list-style-type: none"> - attributeType is at least absent or attributeType = <i>Actual</i>, but <i>Target</i>, <i>MinSet</i>, and <i>MaxSet</i> are also allowed - variable.name = "OfflineThreshold" - component.name = "OCPPCommCtrlr" <p>getVariableData[1]:</p> <ul style="list-style-type: none"> - attributeType is at least absent or attributeType = <i>Actual</i>, but <i>Target</i>, <i>MinSet</i>, and <i>MaxSet</i> are also allowed - variable.name = "AuthorizeRemoteStart" - component.name = "AuthCtrlr"
Post scenario validations: Manually validate that CSMS has correctly read the requested variables.

TC_B_08_CSMS: Get Variables - limit to maximum number of values

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 <code>MaxItemsPerMessageGetVariables</code> .
Purpose	To test that CSMS does not request more variables than the Charging Station reported to support in the variable <code>MaxItemsPerMessageGetVariables</code> .
Prerequisite(s)	N/a

Before (Preparations)

Configuration State: Configure (using <code>getVariablesRequest</code>) Component.Variable.Instance <code>DeviceDataCtrlr.ItemsPerMessage.GetVariables</code> at value 4.
Memory State: N/a
Reusable State(s): N/a

Main (Test scenario)

Charging Station	CSMS
2. Test System 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: - <code>DeviceDataCtrlr.ItemsPerMessage[GetReport]</code> - <code>DeviceDataCtrlr.ItemsPerMessage[GetVariables]</code> - <code>DeviceDataCtrlr.BytesPerMessage[GetReport]</code> - <code>DeviceDataCtrlr.BytesPerMessage[GetVariables]</code> - <code>AuthCtrlr.AuthorizeRemoteStart</code>

Tool validations

<p>* Step 1:</p> <p>Message: GetVariablesRequest for 4 variables and a GetVariablesRequest for 1 variable (in arbitrary order):</p> <p>for component.name = <code>"DeviceDataCtrlr"</code></p> <ul style="list-style-type: none"> - variable.name = <code>"ItemsPerMessage"</code> with variable.instance = <code>"GetReport"</code> - variable.name = <code>"ItemsPerMessage"</code> with variable.instance = <code>"GetVariables"</code> - variable.name = <code>"BytesPerMessage"</code> with variable.instance = <code>"GetReport"</code> - variable.name = <code>"BytesPerMessage"</code> with variable.instance = <code>"GetVariables"</code> <p>and for component.name = <code>"AuthCtrlr"</code></p> <ul style="list-style-type: none"> - variable.name = <code>"AuthorizeRemoteStart"</code>
<p>Post scenario validations:</p> <p>Test System validates that not more than <code>ItemsPerMessageGetVariables</code> elements are requested in one GetVariablesRequest message by CSMS.</p>

TC_B_09_CSMS: Set Variables - single value

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. Test System responds with: SetVariablesResponse	1. <i>Manually request CSMS to set data for:</i> - OCPPCommCtrlr.OfflineThreshold

Tool validations
<p>* Step 1:</p> <p>Message: SetVariablesRequest with (in arbitraty order):</p> <p>setVariableData[1]:</p> <ul style="list-style-type: none"> - variable.name = "OfflineThreshold" - component.name = "OCPPCommCtrlr" - attributeValue = "123" - attributeType is absent or attributeType = <i>Actual</i>
<p>Post scenario validations:</p> <p>Manually validate that CSMS has correctly set the requested variables.</p>

TC_B_10_CSMS: Set Variables - multiple values

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. Test System responds with: SetVariablesResponse	1. <i>Manually request CSMS to set data for:</i> - OCPPCommCtrlr.OfflineThreshold - AuthCtrlr.AuthorizeRemoteStart+

Tool validations
<p>* Step 1:</p> <p>Message: SetVariablesRequest with (in arbitraty order):</p> <p>setVariableData[1]:</p> <ul style="list-style-type: none"> - variable.name = "OfflineThreshold" - component.name = "OCPPCommCtrlr" - attributeValue = "123" - attributeType is absent or attributeType = Actual <p>setVariableData[2]:</p> <ul style="list-style-type: none"> - variable.name = "AuthorizeRemoteStart" - component.name = "AuthCtrlr" - attributeValue = "false" - attributeType is absent or attributeType = Actual <p>Post scenario validations:</p> <p>Manually validate that CSMS has correctly set the requested variables.</p>

TC_B_12_CSMS: Get Base Report - ConfigurationInventory

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. Test System responds with: GetBaseReportResponse	1. <i>Manually instruct CSMS to retrieve a ConfigurationInventory report.</i>

Tool validations
* Step 1: Message: GetBaseReportRequest with: - requestId has integer value ≥ 0 - reportBase = <i>ConfigurationInventory</i>
Post scenario validations: CSMS receives all NotifyReportRequest message for this <i>requestId</i> and is able to show the result of configuration inventory to an operator.

TC_B_13_CSMS: Get Base Report - FullInventory

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. Test System responds with: GetBaseReportResponse	1. <i>Manually instruct CSMS to retrieve a FullInventory report.</i>

Tool validations
* Step 1: GetBaseReportRequest with: - requestId has integer value ≥ 0 - reportBase = <i>FullInventory</i>
Post scenario validations: CSMS receives all NotifyReportRequest message for this <i>requestId</i> and is able to show the result of full inventory to an operator.

TC_B_14_CSMS: Get Base Report - SummaryInventory

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. Test System responds with: GetBaseReportResponse	1. <i>Manually instruct CSMS to retrieve a SummaryInventory report.</i>

Tool validations
* Step 1: GetBaseReportRequest with: - requestId has integer value ≥ 0 - reportBase = <i>SummaryInventory</i>
Post scenario validations: CSMS receives all NotifyReportRequest message for this <i>requestId</i> and is able to show the result of summary inventory to an operator.

TC_B_18_CSMS: Get Custom Report - with componentCriteria and component/variables

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. Test System responds with: GetReportResponse with status <i>EmptyResultSet</i>	1. Manually instruct CSMS to get the value of: - EVSE #1::AvailabilityState - from all <i>Problem</i> components
4. Test System responds with: GetReportResponse with status <i>Accepted</i>	3. Manually instruct CSMS to get the value of: - EVSE #1::AvailabilityState - from all <i>Available</i> components
5. Test System responds with: NotifyReportRequest	6. CSMS sends NotifyReportResponse

Tool validations
* Step 1: Message: GetReportRequest - componentCriteria = <i>Problem</i> - componentVariable[0].component.name = "EVSE" - componentVariable[0].component.evse.id = 1 - componentVariable[0].variable.name = "AvailabilityState"
* Step 3: Message: GetReportRequest - componentCriteria is <i>Available</i> - componentVariable[0].component.name = "EVSE" - componentVariable[0].component.evse.id = 1 - componentVariable[0].variable.name = "AvailabilityState"
Post scenario validations: N/A

TC_B_20_CSMS: Reset Charging Station - Without ongoing transaction - OnIdle

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
<u>Manual Action:</u> Request the CSMS to reboot the Charging Station with type_OnIdle	
2. The Test System responds with a ResetResponse with status <i>Accepted</i>	1. The CSMS sends a ResetRequest
3. The Test System sends a BootNotificationRequest	4. The CSMS responds with a BootNotificationResponse
5. The Test System notifies the CSMS about the current state of all connectors. Message: StatusNotificationRequest - connectorStatus <i>Available</i> Message: NotifyEventRequest - trigger <i>Delta</i> - actualValue <i>"Available"</i> - component.name <i>"Connector"</i> - variable.name <i>"AvailabilityState"</i>	6. The CSMS responds accordingly.

Tool validations
* Step 1: Message ResetRequest - evseld must be omitted
* Step 4: Message BootNotificationResponse - status <i>Accepted</i>
Post scenario validations: - N/a

TC_B_21_CSMS: Reset Charging Station - With Ongoing Transaction - OnIdle

Test case name	Reset Charging Station - With Ongoing Transaction - OnIdle
Test case Id	TC_B_21_CSMS
Use case Id(s)	B12
Requirement(s)	B12.FR.01, B12.FR.03, E07.FR.03
System under test	CSMS
Description	<p>This test case covers how the CSMS can remotely request the Charging Station to reset itself by sending a ResetRequest during a transaction. When ResetRequest "OnIdle" is send the charging stations schedules a reboot after all transactions are stopped.</p> <p>This could for example be necessary if the Charging Station is not functioning correctly.</p>
Purpose	To verify if the CSMS is able to perform the reset mechanism as described at the OCPP specification.
Prerequisite(s)	n/a

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action:</u> Request the CSMS to reboot the Charging Station with status OnIdle	
2. The Test System responds with a ResetResponse with status <i>Scheduled</i>	1. The CSMS sends a ResetRequest with status <i>OnIdle</i>
3. The Test System sends a TransactionEventRequest . - eventType <i>Updated</i> - triggerReason <i>StopAuthorized</i> - transactionInfo.chargingState <i>EVConnected</i> - idToken.idToken <i><Configured valid_idtoken_idtoken></i> - idToken.type <i><Configured valid_idtoken_type></i>	4. The CSMS responds with a TransactionEventResponse .
5. The Test System sends a TransactionEventRequest . - eventType <i>Ended</i> - triggerReason <i>EVCommunicationLost</i> - transactionInfo.chargingState <i>Idle</i> - transactionInfo.stoppedReason <i>EVDisconnected</i>	6. The CSMS responds with a TransactionEventResponse .
7. The Test System sends a BootNotificationRequest with reason <i>ScheduledReset</i>	8. The CSMS responds with a BootNotificationResponse
9. The Test System notifies the CSMS about the current state of all connectors. Message: StatusNotificationRequest - connectorStatus <i>Available</i> Message: NotifyEventRequest - trigger <i>Delta</i> - actualValue <i>"Available"</i> - component.name <i>"Connector"</i> - variable.name <i>"AvailabilityState"</i>	10. The CSMS responds accordingly.

Tool validations
<div>* Step 1: Message ResetRequest - type <i>OnIdle</i> - evseld must be omitted</div> <div>* Step 8: Message BootNotificationResponse - status <i>Accepted</i></div>
<div>Post scenario validations: - N/a</div>

TC_B_22_CSMS: Reset Charging Station - With Ongoing Transaction - Immediate

Test case name	Reset Charging Station - With Ongoing Transaction - Immediate
Test case Id	TC_B_22_CSMS
Use case Id(s)	B12
Requirement(s)	N/a
System under test	CSMS
Description	<p>This test case covers how the CSMS can remotely request the Charging Station to reset itself by sending a ResetRequest during a transaction. When ResetRequest "Immediate" is send the charging stations will try to stop all transactions before rebooting.</p> <p>This could for example be necessary if the Charging Station is not functioning correctly.</p>
Purpose	To verify if the CSMS is able to perform the reset mechanism as described at the OCPP specification.
Prerequisite(s)	n/a

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action:</u> Request the CSMS to reboot the Charging Station with status Immediate	
2. The Test System responds with a ResetResponse with status <i>Accepted</i>	1. The CSMS sends a ResetRequest with status <i>Immediate</i>
3. The Test System sends a TransactionEventRequest . - eventType <i>Ended</i> - triggerReason <i>ResetCommand</i> - transactionInfo.chargingState <i>EVConnected</i> - transactionInfo.stoppedReason <i>ImmediateReset</i> - idToken is omitted	4. The CSMS responds with a TransactionEventResponse .
5. The Test System sends a BootNotificationRequest with reason <i>RemoteReset</i>	6. The CSMS responds with a BootNotificationResponse

Main (Test scenario)	
<p>7. The Test System notifies the CSMS about the current state of all connectors.</p> <p>For <Configured connectorId>:</p> <p>Message: StatusNotificationRequest</p> <ul style="list-style-type: none">- connectorStatus <i>Occupied</i> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none">- trigger <i>Delta</i>- actualValue <i>"Occupied"</i>- component.name <i>"Connector"</i>- variable.name <i>"AvailabilityState"</i> <p>For <Other connector(s)>:</p> <p>Message: StatusNotificationRequest</p> <ul style="list-style-type: none">- connectorStatus <i>Available</i> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none">- trigger <i>Delta</i>- actualValue <i>"Available"</i>- component.name <i>"Connector"</i>- variable.name <i>"AvailabilityState"</i>	<p>8. The CSMS responds accordingly.</p>

Tool validations
<p>* Step 1:</p> <p>Message ResetRequest</p> <ul style="list-style-type: none">- type <i>Immediate</i>- evseld must be omitted <p>* Step 6:</p> <p>Message BootNotificationResponse</p> <ul style="list-style-type: none">- status <i>Accepted</i>
<p>Post scenario validations:</p> <ul style="list-style-type: none">- N/a

TC_B_25_CSMS: Reset EVSE - Without ongoing transaction

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 Test System responds with a ResetResponse with status Accepted	1. The CSMS sends a ResetRequest with status OnIdle and evsID <Configured evseld>

Tool validations
* Step 1: Message ResetRequest - type OnIdle - evseld <Configured evseld>
Post scenario validations: - N/a

TC_B_26_CSMS: Reset EVSE - With Ongoing Transaction - OnIdle

Test case name	Reset EVSE - With Ongoing Transaction - OnIdle
Test case Id	TC_B_26_CSMS
Use case Id(s)	B12
Requirement(s)	B12.FR.07
System under test	CSMS
Description	<p>This test case covers how the CSMS can remotely request the Charging Station to reset an EVSE by sending a ResetRequest during a transaction. When ResetRequest "OnIdle" is send the charging stations schedules a reboot after all transactions are stopped.</p> <p>This could for example be necessary if the Charging Station is not functioning correctly.</p>
Purpose	To verify if the CSMS is able to perform the reset mechanism as described at the OCPP specification.
Prerequisite(s)	n/a

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action:</u> Request the CSMS to reboot the charging EVSE with status OnIdle	
2. The Test System responds with a ResetResponse with status <i>Scheduled</i>	1. The CSMS sends a ResetRequest with status <i>OnIdle</i> and evseld <Configured evseld>
3. The Test System sends a TransactionEventRequest . - eventType <i>Updated</i> - triggerReason <i>StopAuthorized</i> - transactionInfo.chargingState <i>EVConnected</i> - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type>	4. The CSMS responds with a TransactionEventResponse .
5. The Test System sends a TransactionEventRequest . - eventType <i>Ended</i> - triggerReason <i>EVCommunicationLost</i> - transactionInfo.chargingState <i>Idle</i> - transactionInfo.stoppedReason <i>EVDisconnected</i>	6. The CSMS responds with a TransactionEventResponse .

Tool validations
* Step 1: Message ResetRequest - type <i>OnIdle</i> - evseld <Configured evseld>
Post scenario validations: - N/a

TC_B_27_CSMS: Reset EVSE - With Ongoing Transaction - Immediate

Test case name	Reset EVSE - With Ongoing Transaction - Immediate
Test case Id	TC_B_27_CSMS
Use case Id(s)	B12
Requirement(s)	N/a
System under test	CSMS
Description	<p>This test case covers how the CSMS can remotely request the Charging Station to reset an EVSE by sending a ResetRequest during a transaction. When ResetRequest "Immediate" is send the charging stations will try to stop all transactions before rebooting.</p> <p>This could for example be necessary if the Charging Station is not functioning correctly.</p>
Purpose	To verify if the CSMS is able to perform the reset mechanism as described at the OCPP specification.
Prerequisite(s)	n/a

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action:</u> Request the CSMS to reboot the charging EVSE with status Immediate	
2. The Test System responds with a ResetResponse with status <i>Accepted</i>	1. The CSMS sends a ResetRequest with status <i>Immediate</i> and evseld <i><Configured evseld></i>
3. The Test System sends a TransactionEventRequest . - eventType <i>Ended</i> - triggerReason <i>ResetCommand</i> - transactionInfo.chargingState <i>EVConnected</i> - transactionInfo.stoppedReason <i>ImmediateReset</i>	4. The CSMS responds with a TransactionEventResponse .

Tool validations
* Step 1: Message ResetRequest - type <i>Immediate</i> - evseld <i><Configured evseld></i>
Post scenario validations: N/a

TC_B_42_CSMS: Set new NetworkConnectionProfile - Accepted

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 Test System responds with a SetNetworkProfileResponse With status <i>Accepted</i>	1. The CSMS sends a SetNetworkProfileRequest

Tool validations
<p>* Step 1:</p> <p>Message SetNetworkProfileRequest</p> <ul style="list-style-type: none"> - 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

TC_B_44_CSMS: Set new NetworkConnectionProfile - Failed

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 Test System responds with a SetNetworkProfileResponse With status Failed	1. The CSMS sends a SetNetworkProfileRequest

Tool validations
N/a
Post scenario validations: - N/a

TC_B_58_CSMS: WebSocket Subprotocol negotiation

Test case name	WebSocket Subprotocol validation
Test case Id	TC_B_58_CSMS
Use case Id(s)	Part 4 - JSON over WebSockets implementation guide
Requirement(s)	Section 3.1.2. OCPP version
System under test	CSMS
Description	OCPP-J imposes extra constraints on the WebSocket subprotocol, detailed in the following section 3.1.2.
Purpose	To verify whether the CSMS is able to select a supported OCPP version, when also a different unsupported version is supported by the Charging Station and relays this selection via the Sec-Websocket-Protocol header.
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 Test System disconnects the WebSocket connection and reconnects by sending a HTTP upgrade request with the header; <code>Sec-WebSocket-Protocol: ocpp0.1</code>	2. The CSMS rejects the connection attempt and does NOT upgrade the connection to a WebSocket connection.
3. The Test System disconnects the WebSocket connection and reconnects by sending a HTTP upgrade request with the header; <code>Sec-WebSocket-Protocol: ocpp0.1,ocpp<Selected OCPP version></code>	4. The CSMS accepts the connection attempt and upgrades the connection to a WebSocket connection.

Tool validations

* Step 4:

The authorization header of the HTTP upgrade response must contain the header Sec-Websocket-Protocol, and it must comply to the following:

- The header is formatted as follows; `Sec-WebSocket-Protocol: ocpp<Selected OCPP version>`

Post scenario validations:

N/a

TC_B_100_CSMS: Set new NetworkConnectionProfile - Identity and password

Test case name	Set new NetworkConnectionProfile - Identity and password
Test case Id	TC_B_100_CSMS
Use case Id(s)	B09
Requirement(s)	B09.FR.01, B09.FR.14
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 that contains basicAuthPassword and identity at one of the by the Charging Station defined configuration slots.
Prerequisite(s)	CSMS uses the SetNetworkProfileRequest to update a network connection profile.

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

Main (Test scenario)	
Charging Station	CSMS
<u>Note</u> : Request CSMS to send a network profile including identity and password.	
2. The Test System responds with a SetNetworkProfileResponse With status <i>Accepted</i>	1. The CSMS sends a SetNetworkProfileRequest

Tool validations
<p>* Step 1:</p> <p>Message SetNetworkProfileRequest</p> <ul style="list-style-type: none"> - 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> - connectionData.identity <Configured identity> - connectionData.basicAuthPassword <Configured basicAuthPassword>
<p>Post scenario validations:</p> <ul style="list-style-type: none"> - N/a

TC_B_103_CSMS: Reset ImmediateAndResume - With Ongoing Transaction - Resuming Energy Transfer

Test case name	Reset ImmediateAndResume - With Ongoing Transaction - Resuming Energy Transfer
Test case Id	TC_B_103_CSMS
Use case Id(s)	B13
Requirement(s)	B13.FR.31
System under test	CSMS
Description	<p>This test case covers how the CSMS can remotely request the Charging Station to reset itself by sending a ResetRequest during a transaction. When ResetRequest "Immediate" is send the charging stations will try to stop all transactions before rebooting.</p> <p>This could for example be necessary if the Charging Station is not functioning correctly.</p>
Purpose	To verify if the CSMS is able to perform the reset mechanism as described at the OCPP specification.
Prerequisite(s)	n/a

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action:</u> Request the CSMS to reboot the Charging Station with status ImmediateAndResume	
2. The Test System responds with a ResetResponse with status Accepted	1. The CSMS sends a ResetRequest with status ImmediateAndResume
3. The Test System sends a TransactionEventRequest . - eventType Updated - triggerReason ResetCommand - idToken is omitted	4. The CSMS responds with a TransactionEventResponse .
5. The Test System sends a BootNotificationRequest with reason RemoteReset	6. The CSMS responds with a BootNotificationResponse
7. The Test System notifies the CSMS about the current state of all connectors. For <Configured connectorId>: Message: NotifyEventRequest - trigger Delta - actualValue "Occupied" - component.name "Connector" - variable.name "AvailabilityState" For <Other connector(s)>: Message: NotifyEventRequest - trigger Delta - actualValue "Available" - component.name "Connector" - variable.name "AvailabilityState"	8. The CSMS responds accordingly.

Main (Test scenario)	
9 The Test System sends a SecurityEventNotificationRequest with - type <i>ResetOrReboot</i>	10 The CSMS responds with a SecurityEventNotificationResponse
11. The Test System sends a TransactionEventRequest . - eventType <i>Updated</i> - triggerReason <i>TxResumed</i> - transactionInfo.chargingState <i>Charging</i> - idToken is omitted	12. The CSMS responds with a TransactionEventResponse .

Tool validations
* Step 1: Message ResetRequest - type <i>ImmediateAndResume</i> * Step 6: Message BootNotificationResponse - status <i>Accepted</i>
Post scenario validations: - N/a

TC_B_104_CSMS: Reset ImmediateAndResume - Without ongoing transaction

Test case name	Reset ImmediateAndResume - Without ongoing transaction
Test case Id	TC_B_104_CSMS
Use case Id(s)	B13
Requirement(s)	
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)	CSMS has support for resuming transactions

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 the Charging Station with type ImmediateAndResume	
2. The Test System responds with a ResetResponse with status <i>Accepted</i>	1. The CSMS sends a ResetRequest
3. The Test System sends a BootNotificationRequest	4. The CSMS responds with a BootNotificationResponse
5. The Test System notifies the CSMS about the current state of all connectors. Message: NotifyEventRequest - trigger <i>Delta</i> - actualValue <i>"Available"</i> - component.name <i>"Connector"</i> - variable.name <i>"AvailabilityState"</i>	6. The CSMS responds accordingly.

Tool validations
* Step 1: Message ResetRequest - type <i>ImmediateAndResume</i>
* Step 4: Message BootNotificationResponse - status <i>Accepted</i>
Post scenario validations: - N/a

TC_B_105_CSMS: Set new NetworkConnectionProfile - Add new NetworkConfiguration using SetVariables

Test case name	Set new NetworkConnectionProfile - Add new NetworkConfiguration using SetVariables
Test case Id	TC_B_105_CSMS
Use case Id(s)	B09
Requirement(s)	B09.FR.29. B09.FR.30
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 that CSMS can set NetworkConfiguration and correctly sets the value of Apn/VpnEnabled.
Prerequisite(s)	CSMS is able to set the NetworkConfiguration component Test System reports NetworkConfiguration variables as ReadWrite.

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

Main (Test scenario)	
Charging Station	CSMS
<u>Note:</u> In the following steps <configurationSlot> equals <Configured non-active configurationSlot>	
<u>Note:</u> Request CSMS to set a network configuration that includes APN connection details, but no VPN details.	
2. The Charging Station responds with SetVariablesResponse(s) with - setVariableResult.attributeStatus = <i>Accepted</i> for each variable	1. CSMS sends a SetVariablesRequest on component NetworkConfiguration[<Configured non-active configurationSlot>]

Tool validations
<p>* Step 2: (The following can be sent in one or multiple messages.)</p> <p>Message SetVariablesRequest for component NetworkConfiguration[<Configured non-active configurationSlot>] with:</p> <ul style="list-style-type: none"> - OcppCsmsUrl = <Configured ocppCsmsUrl> - OcppInterface = <i>Any</i> - OcppTransport = <i>JSON</i> - OcppVersion = <i>OCPP21</i> - MessageTimeout = <Configured messageTimeout> - SecurityProfile = <i>2</i> - Identity = <Configured charging station identity> - BasicAuthPassword = <i>PasswordOfSufficientLength</i> - VpnEnabled = <i>false</i> - ApnEnabled = <i>true</i> - Apn = <Configured APN URL> - ApnUserName = <Configured APN username> - ApnPassword = <Configured APN password> - ApnAuthentication one of <i>PAP, CHAP, NONE, AUTO</i>
Post scenario validations:
- N/a

TC_B_116_CSMS: Reset ImmediateAndResume - With Ongoing Transaction and SmartCharging - resuming energytransfer

Test case name	Reset ImmediateAndResume - With Ongoing Transaction and SmartCharging - resuming energytransfer
Test case Id	TC_B_116_CSMS
Use case Id(s)	B13
Requirement(s)	B13.FR.31
System under test	CSMS
Description	<p>This test case covers how the CSMS can remotely request the Charging Station to reset itself by sending a ResetRequest during a transaction. When ResetRequest "Immediate" is sent the charging stations will try to stop all transactions before rebooting.</p> <p>This could for example be necessary if the Charging Station is not functioning correctly.</p>
Purpose	To verify if the CSMS is able to perform the reset mechanism as described at the OCPP specification.
Prerequisite(s)	n/a

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Manual action</u> : trigger CSMS to set a Charging Profile of type TxProfile on <transactionId>	
2. The Test System responds with a SetChargingProfileResponse with status <i>Accepted</i>	1. The CSMS sends a SetChargingProfileRequest
<u>Manual Action</u> : Request the CSMS to reboot the Charging Station with status <i>ImmediateAndResume</i>	
4. The Test System responds with a ResetResponse with status <i>Accepted</i>	3. The CSMS sends a ResetRequest
5. The Test System sends a TransactionEventRequest . - eventType <i>Updated</i> - triggerReason <i>ResetCommand</i>	6. The CSMS responds with a TransactionEventResponse .
7. The Test System sends a BootNotificationRequest with reason <i>RemoteReset</i>	8. The CSMS responds with a BootNotificationResponse

Main (Test scenario)	
<p>9. The Test System notifies the CSMS about the current state of all connectors.</p> <p>For <Configured connectorId>:</p> <p>Message: StatusNotificationRequest</p> <ul style="list-style-type: none"> - connectorStatus <i>Occupied</i> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none"> - trigger <i>Delta</i> - actualValue <i>"Occupied"</i> - component.name <i>"Connector"</i> - variable.name <i>"AvailabilityState"</i> <p>For <Other connector(s)>:</p> <p>Message: StatusNotificationRequest</p> <ul style="list-style-type: none"> - connectorStatus <i>Available</i> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none"> - trigger <i>Delta</i> - actualValue <i>"Available"</i> - component.name <i>"Connector"</i> - variable.name <i>"AvailabilityState"</i> 	10. The CSMS responds accordingly.
<p>11. The Test System sends a TransactionEventRequest.</p> <ul style="list-style-type: none"> - eventType <i>Updated</i> - triggerReason <i>TxResumed</i> - idToken is omitted 	12. The CSMS responds with a TransactionEventResponse .
<p>14. The Test System responds with a SetChargingProfileResponse with status <i>Accepted</i></p>	13. The CSMS sends a SetChargingProfileRequest

Tool validations
<p>* Step 1:</p> <p>Message SetChargingProfileRequest</p> <ul style="list-style-type: none"> - chargingProfile.chargingProfilePurpose <i>TxProfile</i> <p>* Step 3:</p> <p>Message ResetRequest</p> <ul style="list-style-type: none"> - type <i>ImmediateAndResume</i> <p>* Step 8:</p> <p>Message BootNotificationResponse</p> <ul style="list-style-type: none"> - status <i>Accepted</i> <p>* Step 13:</p> <p>Message: SetChargingProfileRequest must either contain an identical <i>chargingProfile</i> as in Step 1 or identical from the time of step 11 onwards.</p>
<p>Post scenario validations:</p> <ul style="list-style-type: none"> - N/a

C Authorization

TC_C_02_CSMS: Local start transaction - Authorization Invalid/Unknown

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 Test System 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 <i>Invalid</i> or <i>Unknown</i>
Post scenario validations: - N/a

TC_C_06_CSMS: Local start transaction - Authorization Blocked

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 Test System, 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 Test System 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 <i>Blocked</i> or <i>Invalid</i>
Post scenario validations:

TC_C_07_CSMS: Local start transaction - Authorization Expired

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 Test System, 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 Test System 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:

TC_C_08_CSMS: Authorization through authorization cache - Accepted

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
Reusable State: State is <i>EVConnectedPreSession</i>

Main (Test scenario)	
Charging Station	CSMS
1. The Test System sends a TransactionEventRequest with - triggerReason <i>Authorized</i> - idToken <i><Valid id token configured in Authorization Cache></i> - eventType <i>Updated</i> <u>Note(s):</u> - TxStartPoint contains <i>ParkingBayOccupancy</i>	2. The CSMS responds with a TransactionEventResponse

Tool validations
* Step 2: Message TransactionEventResponse - idTokenInfo.status <i>Accepted</i>
Post scenario validations: - N/a

TC_C_20_CSMS: Authorization through authorization cache - Invalid

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
Reusable State: State is <i>EVConnectedPreSession</i>

Main (Test scenario)	
Charging Station	CSMS
1. The Test System sends a TransactionEventRequest with - triggerReason <i>Authorized</i> - idToken.idToken <i><Configured invalid_idtoken_idtoken></i> - idToken.type <i><Configured invalid_idtoken_type></i> - eventType <i>Updated</i> <u>Note(s):</u> - TxStartPoint contains <i>ParkingBayOccupancy</i>	2. The CSMS responds with a TransactionEventResponse

Tool validations
* Step 2: Message TransactionEventResponse - idTokenInfo.status <i>Invalid</i> or <i>Unknown</i>
Post scenario validations: - N/a

TC_C_37_CSMS: Clear Authorization Data in Authorization Cache - Accepted

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 Test System responds with a ClearCacheResponse with status <i>Accepted</i>	1. The CSMS sends a ClearCacheRequest

Tool validations
- N/a
Post scenario validations: - N/a

TC_C_38_CSMS: Clear Authorization Data in Authorization Cache - Rejected

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 Test System responds with a ClearCacheResponse with status <i>Rejected</i>	1. The CSMS sends a ClearCacheRequest

Tool validations
- N/a
Post scenario validations: - N/a

TC_C_39_CSMS: Authorization by GroupId - Success

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 Test System 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 Test System sends a TransactionEventRequest with - triggerReason <i>Authorized</i> - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type> if transaction was already started - eventType <i>Updated</i> else - eventType <i>Started</i>	4. The CSMS responds with a TransactionEventResponse
5. Execute Reusable State EnergyTransferStarted	
6. The Test System 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 Test System sends a TransactionEventRequest with - triggerReason <i>StopAuthorized</i> - idToken.idToken <Configured valid_idtoken2_idtoken> - idToken.type <Configured valid_idtoken2_type> - eventType <i>Updated</i>	9. The CSMS responds with a TransactionEventResponse
10. Execute Reusable State EVConnectedPostSession	
11. Execute Reusable State EVDisconnected	

Tool validations
<p>* Step 2:</p> <p>Message AuthorizeResponse</p> <ul style="list-style-type: none">- idTokenInfo.status <i>Accepted</i>- idTokenInfo.groupIdToken.idToken <i><Configured groupIdToken></i> <p>* Step 4:</p> <p>Message TransactionEventResponse</p> <ul style="list-style-type: none">- idTokenInfo.status <i>Accepted</i>- idTokenInfo.groupIdToken.idToken <i><Configured groupIdToken></i> <p>* Step 7:</p> <p>Message AuthorizeResponse</p> <ul style="list-style-type: none">- idTokenInfo.status <i>Accepted</i>- idTokenInfo.groupIdToken.idToken <i><Configured groupIdToken></i> <p>* Step 9:</p> <p>Message TransactionEventResponse</p> <ul style="list-style-type: none">- idTokenInfo.status <i>Accepted</i>- idTokenInfo.groupIdToken.idToken <i><Configured groupIdToken></i>
<p>Post scenario validations:</p> <ul style="list-style-type: none">- N/a

TC_C_40_CSMS: Authorization by GroupId - Success with Local Authorization List

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 Test System sends a TransactionEventRequest with - triggerReason <i>Authorized</i> - idToken.idToken <i><Configured valid_idtoken_idtoken></i> (with a configured GroupId) which is configured in the local Authorization List - idToken.type <i><Configured valid_idtoken_type></i> (with a configured GroupId) which is configured in the local Authorization List If transaction was already started - eventType <i>Updated</i> else - eventType <i>Started</i>	2. The CSMS responds with a TransactionEventResponse
3. Execute Reusable State <i>EnergyTransferStarted</i>	
5. The Test System sends a TransactionEventRequest with - triggerReason <i>StopAuthorized</i> - idToken.idToken <i><Configured valid_idtoken2_idtoken></i> (with same configured GroupId) which is configured in the local Authorization List - idToken.type <i><Configured valid_idtoken2_type></i> - eventType <i>Updated</i>	6. The CSMS responds with a TransactionEventResponse
7. Execute Reusable State <i>EVConnectedPostSession</i>	
8. Execute Reusable State <i>EVDisconnected</i>	

Tool validations
<p>* Step 2:</p> <p>Message TransactionEventResponse</p> <ul style="list-style-type: none">- idTokenInfo.status <i>Accepted</i>- idTokenInfo.groupIdToken.idToken <i><Configured groupIdToken></i> <p>* Step 6:</p> <p>Message TransactionEventResponse</p> <ul style="list-style-type: none">- idTokenInfo.status <i>Accepted</i>- idTokenInfo.groupIdToken.idToken <i><Configured groupIdToken></i>
<p>Post scenario validations:</p> <ul style="list-style-type: none">- N/a

TC_C_43_CSMS: Authorization by GroupId - Invalid status with Local Authorization List

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 Test System sends a TransactionEventRequest with - triggerReason <i>Authorized</i> - idToken.idToken <i><Configured valid_idtoken_idtoken></i> - idToken.type <i><Configured valid_idtoken_type></i> if transaction was already started - eventType <i>Updated</i> else - eventType <i>Started</i>	2. The CSMS responds with a TransactionEventResponse
3. Execute Reusable State EnergyTransferStarted	
4. The Test System sends an AuthorizeRequest with - idToken.idToken <i><Configured valid_idtoken2_idtoken></i> - idToken.type <i><Configured valid_idtoken2_type></i>	5. The CSMS responds with an AuthorizeResponse
6. The Test System sends a TransactionEventRequest with - triggerReason <i>StopAuthorized</i> - idToken.idToken <i><Configured valid_idtoken2_idtoken></i> - idToken.type <i><Configured valid_idtoken2_type></i> - eventType <i>Updated</i>	7. The CSMS responds with a TransactionEventResponse
8. Execute Reusable State EVConnectedPostSession	
9. Execute Reusable State EVDisconnected	

Tool validations
<p>* Step 1:</p> <p>Message TransactionEventResponse</p> <ul style="list-style-type: none">- idTokenInfo.status <i>Accepted</i>- idTokenInfo.groupIdToken.idToken <i><Configured groupIdToken></i> <p>* Step 5:</p> <p>Message AuthorizeResponse</p> <ul style="list-style-type: none">- idTokenInfo.status <i>Accepted</i>- idTokenInfo.groupIdToken.idToken <i><Configured groupIdToken></i> <p>* Step 7:</p> <p>Message TransactionEventResponse</p> <ul style="list-style-type: none">- idTokenInfo.status <i>Accepted</i>- idTokenInfo.groupIdToken.idToken <i><Configured groupIdToken></i>
<p>Post scenario validations:</p> <ul style="list-style-type: none">- N/a

TC_C_47_CSMS: Stop Transaction with a Master Pass - With UI - All transactions

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 idTokenState is *EnergyTransferStarted* for EVSE 2 with idToken valid idToken2

Main (Test scenario)

Charging Station	CSMS
1. The Test System 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 Test System 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

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

TC_C_48_CSMS: Stop Transaction with a Master Pass - With UI - With UI - Specific transactions

Test case name	Stop Transaction with a Master Pass - With UI - With UI - Specific transactions
Test case Id	TC_C_48_CSMS
Use case Id(s)	C16
Requirement(s)	C16_FR_01
System under test	CSMS
Description	This test case covers how somebody with a Master Pass (User) can stop (selected) ongoing transactions, so the cable becomes unlocked. This Master Pass can be configured in: MasterPassGroupId. This could for example be usefull for Law Enforcement officials.
Purpose	To verify if the CSMS is able to correctly respond on a request to stop a transaction when an idToken which has the MasterPass as GroupId is used and the user has selected to stop one transaction in the User Interface according to the mechanism as described in the OCPP specification.
Prerequisite(s)	- N/a

Before (Preparations)
Configuration State: N/a
Memory State: An idToken with the MastersPass as GroupId is configured
Reusable State(s): State is <i>EnergyTransferStarted</i> for all EVSE

Main (Test scenario)	
Charging Station	CSMS
1. The Test System 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 Test System 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
<p>* Step 2:</p> <p>Message AuthorizeResponse</p> <ul style="list-style-type: none"> - idTokenInfo.status Accepted - idTokenInfo.groupIdToken.idToken <Configured masterPassGroupId> <p>* Step 4:</p> <p>Message TransactionEventResponse</p> <ul style="list-style-type: none"> - idTokenInfo.status Accepted - idTokenInfo.groupIdToken.idToken <Configured masterPassGroupId>
Post scenario validations: - N/a

TC_C_49_CSMS: Stop Transaction with a Master Pass - Without UI

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 <i>EnergyTransferStarted</i> for EVSE 1 with idToken valid idToken State is <i>EnergyTransferStarted</i> for EVSE 2 with idToken valid idToken2

Main (Test scenario)	
Charging Station	CSMS
1. The Test System 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 Test System 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

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

TC_C_50_CSMS: Authorization using Contract Certificates 15118 - Online - Local contract certificate validation - Accepted

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)	<ul style="list-style-type: none"> - The configured eMAID is known by the CSMS as valid. - The configured contract certificate is valid. - The CN of the configured contract certificate equals the configured eMAID. - iso15118CertificateHashData has a responder URL that points to an OCSP service for Test System. - 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 Test System sends an AuthorizeRequest With idToken.idToken <Configured eMAID> idToken.type eMAID 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 Test System OCSP service responds that certificate is valid.	4. The CSMS responds with a AuthorizeResponse
5. The Test System 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: Test System 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

TC_C_51_CSMS: Authorization using Contract Certificates 15118 - Online - Local contract certificate validation - Rejected

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)	<ul style="list-style-type: none"> - 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 Test System. - 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 <i>EVConnectedPreSession</i>

Main (Test scenario)	
Charging Station	CSMS
1. The Test System 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 Test System OCSP service responds that certificate is valid.	4. The CSMS responds with a AuthorizeResponse

Tool validations
* Step 2: CSMS sends an OCSP request for iso15118CertificateHashData * Step 3: Test System checks that received request for iso15118CertificateHashData is valid * Step 4: Message: AuthorizeResponse - idTokenInfo.status <i>Invalid</i> - certificateStatus <i>CertificateRevoked</i>
Post scenario validations: EV is not authorized and shall not charge: Charging Station does not send TransactionEventRequest with: - triggerReason = <i>Authorized</i> or chargingState = <i>Charging</i>

TC_C_52_CSMS: Authorization using Contract Certificates 15118 - Online - Central contract certificate validation - Accepted

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)	<ul style="list-style-type: none"> - The configured eMAID is known by the CSMS as valid. - The 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 Test System. - 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 <i>EVConnectedPreSession</i>

Main (Test scenario)	
Charging Station	CSMS
1. The Test System sends an AuthorizeRequest With idToken.idToken <Configured valid_idtoken_idtoken> idToken.type <Configured valid_idtoken_type> iso15118CertificateHashData is absent certificate from keystore	2. The CSMS sends an OCSP request to responder URL of certificate to check validity
3. The Test System OCSP service responds that certificate is valid.	4. The CSMS responds with a AuthorizeResponse
5. The Test System sends a TransactionEventRequest With triggerReason <i>Authorized</i>	6. The CSMS responds with a TransactionEventResponse
5. Execute Reusable State <i>EnergyTransferStarted</i>	

Tool validations
* Step 2: CSMS sends an OCSP request for certificate * Step 3: Test System checks that received request for certificate is valid AND key type = ECDSA AND certificate chain contains at least one subCA * Step 4: Message: AuthorizeResponse - idTokenInfo.status <i>Accepted</i> - certificateStatus <i>Accepted</i> * Step 6: Message: TransactionEventResponse - idTokenInfo.status <i>Accepted</i>
Post scenario validations: N/a

TC_C_103_CSMS: Authorization with prepaid card - success

Test case name	Authorization with prepaid card - success
Test case Id	TC_C_103_CSMS
Use case Id(s)	C17
Requirement(s)	C17.FR.01, C17.FR.03
System under test	CSMS
Description	This test case verifies if the CSMS is able to authorize a prepaid card for a transaction.
Purpose	To verify if the CSMS is able to authorize a prepaid card for a transaction.
Prerequisite(s)	- CSMS supports prepaid cards. - Prepaid card with positive amount is available.

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

Main (Test scenario)	
Charging Station	CSMS
1. The Test System sends a AuthorizeRequest with idToken.idToken <prepaid card id> idToken.type = <prepaid card id type>	2. The CSMS responds a AuthorizeResponse
3. The Test System sends a TransactionEventRequest with - eventType Started - idToken.idToken <prepaid card id> - idToken.type = <prepaid card id type>	4. The CSMS responds with a TransactionEventResponse

Tool validations
<p>* Step 2: Message AuthorizeResponse - idTokenInfo.status Accepted - idTokenInfo.cacheExpiryDateTime <current date/time></p> <p>* Step 4: Message TransactionEventResponse - idTokenInfo.status Accepted - idTokenInfo.cacheExpiryDateTime <cacheExpiryDateTime from step 2> - transactionLimit.maxCost = <not omitted></p>

TC_C_104_CSMS: Authorization with prepaid card - no credit

Test case name	Authorization with prepaid card - no credit
Test case Id	TC_C_104_CSMS
Use case Id(s)	C17
Requirement(s)	C17.FR.02
System under test	CSMS
Description	This test case verifies if the CSMS is able to authorize a prepaid card for a transaction.
Purpose	To verify if the CSMS is able to reject a prepaid card for a transaction.
Prerequisite(s)	- CSMS supports prepaid cards. - Prepaid card with an amount of "0" is available.

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

Main (Test scenario)	
Charging Station	CSMS
1. The Test System sends a AuthorizeRequest with idToken.idToken <prepaid card no credit id> idToken.type = <prepaid card no credit id type>	2. The CSMS responds a AuthorizeResponse

Tool validations
* Step 2: Message AuthorizeResponse - idTokenInfo.status NoCredit - idTokenInfo.cacheExpiryDateTime <current date/time>

TC_C_108_CSMS: Integrated Payment Terminal - VAT number validation

Test case name	Integrated Payment Terminal - VAT number validation
Test case Id	TC_C_108_CSMS
Use case Id(s)	C18
Requirement(s)	C18.FR.09, C18.FR.10
System under test	CSMS
Description	To start/authorize a transaction from a payment terminal connected directly to the Charging Station.
Purpose	To verify that the CSMS can validate VAT numbers correctly
Prerequisite(s)	* CSMS supports VAT number validation

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

Main (Test scenario)	
Charging Station	CSMS
<i>Valid VAT number with EVSE ID</i>	
1. The Test System sends a VatNumberValidationRequest with vatNumber = <validVatNumber> evseld = 1	2. The CSMS validates given VAT number and responds with VatNumberValidationResponse
<i>Invalid VAT number with EVSE ID</i>	
3. The Test System sends a VatNumberValidationRequest with vatNumber = <invalidVatNumber> evseld = 1	4. The CSMS validates given VAT number and responds with VatNumberValidationResponse
<i>Valid VAT number without EVSE ID</i>	
5. The Test System sends a VatNumberValidationRequest with vatNumber = <validVatNumber> evseld = <omitted>	6. The CSMS validates given VAT number and responds with VatNumberValidationResponse
<i>Invalid VAT number without EVSE ID</i>	
7. The Test System sends a VatNumberValidationRequest with vatNumber = <invalidVatNumber> evseld = <omitted>	8. The CSMS validates given VAT number and responds with VatNumberValidationResponse

Tool validations
<p>* Step 2:</p> <p>Message VatNumberValidationResponse</p> <ul style="list-style-type: none">- vatNumber must be <i><validVatNumber></i>- evseld must be <i>1</i>- status must be <i>Accepted</i>
<p>* Step 4:</p> <p>Message VatNumberValidationResponse</p> <ul style="list-style-type: none">- vatNumber must be <i><invalidVatNumber></i>- evseld must be <i>1</i>- status must be <i>Rejected</i>
<p>* Step 6:</p> <p>Message VatNumberValidationResponse</p> <ul style="list-style-type: none">- vatNumber must be <i><validVatNumber></i>- evseld must be <i><omitted></i>- status must be <i>Accepted</i>
<p>* Step 8:</p> <p>Message VatNumberValidationResponse</p> <ul style="list-style-type: none">- vatNumber must be <i><invalidVatNumber></i>- evseld must be <i><omitted></i>- status must be <i>Rejected</i>
<p>Post scenario validations:</p> <ul style="list-style-type: none">- N/a

TC_C_113_CSMS: Integrated Payment Terminal - Cancellation after start of transaction - stopped by EV driver

Test case name	Integrated Payment Terminal - Cancellation after start of transaction - stopped by EV driver
Test case Id	TC_C_113_CSMS
Use case Id(s)	C20
Requirement(s)	C20.FR.05
System under test	CSMS
Description	To inform the CSMS that payment has been canceled and the authorization amount has been released.
Purpose	To verify if the CSMS is able to handle Cancellation of a local payment terminal authorization after a transaction has started when the EV driver stops the session at the Charging Station.
Prerequisite(s)	<ul style="list-style-type: none"> - CSMS supports Charging Stations with integrated payment terminals. - Test System has PaymentCtrlr.AuthorizeDirectPayment = <i>false</i> - Test System has PaymentCtrlr.AuthorizationAmount = <i>50.00</i> - Test System has PaymentCtrlr.PaymentDetails = "CardLast4Digits" - Test System has PaymentCtrlr.SettlementByCSMS = <i>false</i>

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

Main (Test scenario)	
Charging Station	CSMS
<u>Note:</u> Starts and immediately ends a transaction	
1. The Test System sends a TransactionEventRequest with eventType <i>Started</i> triggerReason <i>Authorized</i> idToken <i><pspRef></i> idToken.type <i>DirectPayment</i> idToken.additionalInfo[0].additionalIdToken = <i>1234</i> idToken.additionalInfo[0].type = <i>Last4Digits</i> transactionInfo.transactionId <i><transactionId></i> transactionInfo.transactionLimit.maxCost <i>50.00</i>	2. The CSMS responds with a TransactionEventResponse
3. The Test System sends a TransactionEventRequest with eventType <i>Ended</i> idToken <i><pspRef></i> idToken.type <i>DirectPayment</i> idToken.additionalInfo[0].additionalIdToken = <i>1234</i> idToken.additionalInfo[0].type = <i>Last4Digits</i> transactionInfo.transactionId <i><transactionId></i> transactionInfo.stoppedReason <i>Local</i> triggerReason <i>StopAuthorized</i> costDetails.totalCost.currency <i>EUR</i> costDetails.totalCost.typeOfCost <i>NormalCost</i> costDetails.totalCost.total <i>0</i> costDetails.totalUsage.energy <i>0</i> costDetails.totalUsage.chargingTime <i>0</i> costDetails.totalUsage.idleTime <i>0</i>	4. The CSMS responds with a TransactionEventResponse

Main (Test scenario)	
5. The Test System sends a NotifySettlementRequest with transactionId <transactionId> pspRef <pspRef> status <i>Canceled</i> settlementTime <Equal to timestamp step 3> settlementAmount 0 receiptUrl <omitted>	6. The CSMS responds with a NotifySettlementResponse

Tool validations
N/a
Post scenario validations: N/a

TC_C_117_CSMS: Settlement at end of transaction - settled by CSMS

Test case name	Settlement at end of transaction - settled by CSMS, receipt by CSMS
Test case Id	TC_C_117_CSMS
Use case Id(s)	C21
Requirement(s)	C21.FR.03, C21.FR.05
System under test	CSMS
Description	Settle the amount of charging session via CSMS.
Purpose	To verify if the CSMS is able to settle a transaction directly via PSP.
Prerequisite(s)	<ul style="list-style-type: none"> - CSMS has an interface to the PSP. - CSMS supports Charging Stations with integrated payment terminals. - Test System has PaymentCtrlr.Enabled = <i>true</i> - Test System has PaymentCtrlr.AuthorizeDirectPayment = <i>false</i> - Test System has PaymentCtrlr.AuthorizeAmount = <i>50.00</i> - Test System has PaymentCtrlr.PaymentDetails = "CardLast4Digits" - Test System has PaymentCtrlr.SettlementByCSMS = <i>true</i> - Value of PaymentCtrlr.ReceiptByCSMS is irrelevant.

Before (Preparations)

Configuration State:

N/a

Memory State:

N/a

Reusable State(s):

N/a

Main (Test scenario)

Charging Station	CSMS
1. The Test System sends a TransactionEventRequest with eventType <i>Started</i> idToken <PspRef> idToken.type <i>DirectPayment</i> transactionInfo.transactionId <transactionId> idToken.additionalInfo[0].additionalIdToken = <not omitted> idToken.additionalInfo[0].type = <i>CardLast4Digits</i> transactionInfo.transactionLimit.maxCost <i>50.00</i>	2. The CSMS responds with a TransactionEventResponse
3. The Test System sends a TransactionEventRequest with eventType <i>Ended</i> idToken <PspRef> idToken.type <i>DirectPayment</i> triggerReason <i>StopAuthorized</i> transactionInfo.stoppedReason <i>Local</i> transactionInfo.transactionId <transactionId> costDetails.totalCost.currency <i>EUR</i> costDetails.totalCost.fixed.exclTax <i>15.00</i> costDetails.totalCost.fixed.inclTax <i>18.15 EUR</i> costDetails.totalCost.fixed.taxRate.type <i>MyTax</i> costDetails.totalCost.fixed.taxRate.tax <i>21</i> costDetails.totalUsage.energy <i>123</i> costDetails.totalUsage.chargingTime <i>5</i> costDetails.totalUsage.idleTime <i>0</i> costDetails.chargingPeriods <omitted>	4. The CSMS responds with a TransactionEventResponse

Note(s):

- The CSMS settles the total cost of the transaction with the payment service provider for amount of 18.15 EUR.

Tool validations
N/a
Post scenario validations:
N/a

TC_C_118_CSMS: Settlement at end of transaction - settled by CS, receipt by CSMS

Test case name	Settlement at end of transaction - settled by CS, receipt by CSMS
Test case Id	TC_C_118_CSMS
Use case Id(s)	C21
Requirement(s)	C21.FR.03
System under test	CSMS
Description	In order to test that Charging Station supports settlement at the end of a transaction where the settlement is handled by the Charging Station and the receipt is provided by the CSMS.
Purpose	To verify that the CSMS can properly handle a transaction where the settlement is done by the Charging Station and the CSMS provides the receipt.
Prerequisite(s)	<ul style="list-style-type: none"> - CSMS supports settlement at the end of a transaction. - CSMS supports Charging Stations with integrated payment terminals. - Test System has PaymentCtrlr.Enabled = <i>true</i> - Test System has PaymentCtrlr.AuthorizeDirectPayment = <i>false</i> - Test System has PaymentCtrlr.AuthorizationAmount = <i>50.00</i> - Test System has PaymentCtrlr.PaymentDetails = "CardLast4Digits" - Test System has PaymentCtrlr.SettlementByCSMS = <i>false</i> - Test System has PaymentCtrlr.ReceiptByCSMS = <i>true</i> - CSMS lets the CS handle the settlement at the end of a transaction, but the CSMS provides receipts.

Before (Preparations)

Configuration State:

N/a

Memory State:

N/a

Reusable State(s):

N/a

Main (Test scenario)

Charging Station	CSMS
<p>1. The Test System sends a TransactionEventRequest with eventType <i>Started</i></p> <p>idToken <configured PspRef></p> <p>idToken.type <i>DirectPayment</i></p> <p>transactionInfo.transactionId <transactionId></p> <p>idToken.additionalInfo[0].additionalIdToken = <not omitted></p> <p>idToken.additionalInfo[0].type = <i>CardLast4Digits</i></p> <p>transactionInfo.transactionLimit.maxCost <i>50.00</i></p>	<p>2. The CSMS responds with a TransactionEventResponse</p>

Main (Test scenario)	
3. The Test System sends a TransactionEventRequest with eventType <i>Ended</i> triggerReason <i>StopAuthorized</i> idToken <i><configured PspRef></i> idToken.type <i>DirectPayment</i> transactionInfo.transactionId <i><transactionId></i> transactionInfo.stoppedReason <i>Local</i> costDetails.totalCost.currency <i>EUR</i> costDetails.totalCost.fixed.exclTax <i>15.00</i> costDetails.totalCost.fixed.inclTax <i>18.15 EUR</i> costDetails.totalCost.fixed.taxRate.type <i>MyTax</i> costDetails.totalCost.fixed.taxRate.tax <i>21</i> costDetails.totalUsage.energy <i>123</i> costDetails.totalUsage.chargingTime <i>5</i> costDetails.totalUsage.idleTime <i>0</i> costDetails.chargingPeriods <i><omitted></i>	4. The CSMS responds with a TransactionEventResponse
5. The Test System sends a NotifySettlementRequest with transactionId <i><transactionId></i> pspRef <i><configured PspRef></i> status <i>Settled</i> settlementTime <i><not omitted></i> settlementAmount <i>18.15</i> receiptUrl <i><omitted></i>	6. The CSMS responds with a NotifySettlementResponse

Tool validations
* Step 6: Message NotifySettlementResponse - receiptUrl must be <i><not omitted></i>
Post scenario validations: - Verify that the receipt URL provided by the CSMS is valid and accessible.

TC_C_119_CSMS: Settlement - is rejected or fails - Failed

Test case name	Settlement - is rejected or fails - Failed
Test case Id	TC_C_119_CSMS
Use case Id(s)	C22
Requirement(s)	...
System under test	CSMS
Description	To inform the CSMS that the transaction settlement has been rejected or otherwise failed to be successfully settled.
Purpose	To verify if the CSMS is able handle if the settlement failed according the Charging Station.
Prerequisite(s)	<ul style="list-style-type: none"> - CSMS supports Charging Stations with integrated payment terminals. - Test System has PaymentCtrlr.Enabled is <i>true</i> - Test System has PaymentCtrlr.AuthorizationAmount = 50.00 - Test System has PaymentCtrlr.PaymentDetails = "CardLast4Digits" - Test System has PaymentCtrlr.SettlementByCSMS is <i>false</i> - Test System has PaymentCtrlr.AuthorizeDirectPayment is <i>false</i>

Before (Preparations)

Configuration State:

N/a

Memory State:

N/a

Reusable State(s):

N/a

Main (Test scenario)

Charging Station	CSMS
1. The Test System sends a TransactionEventRequest with eventType <i>Started</i> idToken <configured PspRef> idToken.type <i>DirectPayment</i> transactionInfo.transactionId <transactionId> idToken.additionalInfo[0].additionalIdToken = <not omitted> idToken.additionalInfo[0].type = <i>CardLast4Digits</i> transactionInfo.transactionLimit.maxCost 50.00	2. The CSMS responds with a TransactionEventResponse
3. The Test System sends a TransactionEventRequest with eventType <i>Ended</i> idToken <configured PspRef> idToken.type <i>DirectPayment</i> transactionInfo.transactionId <transactionId> costDetails.totalCost.currency <i>EUR</i> costDetails.totalCost.fixed.exclTax 15.00 costDetails.totalCost.fixed.inclTax 18.15 costDetails.totalCost.fixed.taxRate.type <i>MyTax</i> costDetails.totalCost.fixed.taxRate.tax 21 costDetails.totalUsage.energy 123 costDetails.totalUsage.chargingTime 5 costDetails.totalUsage.idleTime 0 costDetails.chargingPeriods <omitted>	4. The CSMS responds with a TransactionEventResponse

Main (Test scenario)	
5. The Test System sends a NotifySettlementRequest with transactionId <transactionId> pspRef <configured PspRef> status Failed settlementTime <not omitted> settlementAmount 18.15 receiptUrl <omitted>	6. The CSMS responds with a NotifySettlementResponse

Tool validations
N/a
Post scenario validations:
N/a

TC_C_120_CSMS: Settlement - is rejected or fails - Rejected

Test case name	Settlement - is rejected or fails - Rejected
Test case Id	TC_C_120_CSMS
Use case Id(s)	C22
Requirement(s)	...
System under test	CSMS
Description	To inform the CSMS that the transaction settlement has been rejected or otherwise failed to be successfully settled.
Purpose	To verify if the CSMS is able to handle if the settlement is rejected according the Charging Station.
Prerequisite(s)	<ul style="list-style-type: none"> - CSMS supports Charging Stations with integrated payment terminals. - Test System has PaymentCtrlr.Enabled is <i>true</i> - Test System has PaymentCtrlr.AuthorizationAmount = 50.00 - Test System has PaymentCtrlr.PaymentDetails = "CardLast4Digits" - Test System has PaymentCtrlr.SettlementByCSMS is <i>false</i> - Test System has PaymentCtrlr.ReceiptByCSMS is <i>false</i> - Test System has PaymentCtrlr.AuthorizeDirectPayment is <i>false</i>

Before (Preparations)

Configuration State:

N/a

Memory State:

N/a

Reusable State(s):

N/a

Main (Test scenario)

Charging Station	CSMS
1. The Test System sends a TransactionEventRequest with eventType <i>Started</i> idToken <configured PspRef> idToken.type <i>DirectPayment</i> transactionInfo.transactionId <transactionId> idToken.additionalInfo[0].additionalIdToken = <not omitted> idToken.additionalInfo[0].type = <i>CardLast4Digits</i> transactionInfo.transactionLimit.maxCost 50.00	2. The CSMS responds with a TransactionEventResponse
3. The Test System sends a TransactionEventRequest with eventType <i>Ended</i> idToken <configured PspRef> idToken.type <i>DirectPayment</i> transactionInfo.transactionId <transactionId> costDetails.totalCost.currency <i>EUR</i> costDetails.totalCost.fixed.exclTax 15.00 costDetails.totalCost.fixed.inclTax 18.15 <i>EUR</i> costDetails.totalCost.fixed.taxRate.type <i>MyTax</i> costDetails.totalCost.fixed.taxRate.tax 21 costDetails.totalUsage.energy 123 costDetails.totalUsage.chargingTime 5 costDetails.totalUsage.idleTime 0 costDetails.chargingPeriods is <omitted>	4. The CSMS responds with a TransactionEventResponse

Main (Test scenario)	
5. The Test System sends a NotifySettlementRequest with transactionId <transactionId> pspRef <configured PspRef> status <i>Rejected</i> settlementTime <not omitted> settlementAmount 18.15 receiptUrl <omitted>	6. The CSMS responds with a NotifySettlementResponse

Tool validations
N/a
Post scenario validations:
N/a

TC_C_125_CSMS: Ad hoc payment via stand-alone payment terminal - central cost calculation

Test case name	Ad hoc payment via stand-alone payment terminal - central cost calculation
Test case Id	TC_C_125_CSMS
Use case Id(s)	C24
Requirement(s)	C24.FR.01, C24.FR.02, C24.FR.10
System under test	CSMS
Description	In order to test that Charging Station supports ad hoc payment via a stand-alone payment terminal with central cost calculation.
Purpose	To verify that the CSMS can properly handle ad hoc payments made via a stand-alone payment terminal when using central cost calculation.
Prerequisite(s)	<ul style="list-style-type: none"> - CSMS is connected with a stand-alone payment terminal - Test System has AuthCtrlr.AuthorizeRemoteStart = <i>false</i> - Test System has TariffCostCtrlr.Enabled[Cost] = <i>false</i>

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> : Present a payment card to the payment terminal Kiosk for Charging Station with identity <Configured unique Charging Station identifier>	
2. The Test System responds with a RequestStartTransactionResponse with status <i>Accepted</i>	1. The CSMS sends a RequestStartTransactionRequest
3. The Test System sends a TransactionEventRequest with eventType <i>Started</i> idToken <idToken from step 1> idToken.type <i>DirectPayment</i> idToken.additionalInfo <additionalInfo> transactionInfo.transactionId <transactionId> meterValue[0].timestamp <not omitted> meterValue[0].sampledValue[0].value <i>10000</i> meterValue[0].sampledValue[0].context <i>Transaction.Begin</i>	4. The CSMS responds with a TransactionEventResponse
Settlement	
5. The Test System sends a TransactionEventRequest with eventType <i>Ended</i> idToken <idToken from step 1> idToken.type <i>DirectPayment</i> transactionInfo.transactionId <transactionId> transactionInfo.transactionLimit.maxCost <not omitted> meterValue[0].timestamp <not omitted> meterValue[0].sampledValue[0].value <i>15000</i> meterValue[0].sampledValue[0].context <i>Transaction.End</i>	6. The CSMS responds with a TransactionEventResponse
<u>Note(s)</u> : - The CSMS sends a message to Payment Kiosk or PSP to settle the cost for amount calculated for 5000 Wh	

Tool validations
<p>* Step 1:</p> <p>Message RequestStartTransactionRequest</p> <ul style="list-style-type: none">- evseld must be <i><evse></i>- remoteStartId must be <i><not omitted></i>- idToken.idToken <i><Not empty></i>- idToken.type must be <i>DirectPayment</i>- idToken.additionalInfo.additionalIdToken must be <i><Configured CardLast4Digits></i>- idToken.additionalInfo.type must be <i>CardLast4Digits</i>
<p>* Step 4:</p> <p>Message TransactionEventResponse</p> <ul style="list-style-type: none">- transactionLimit.maxCost <i><not omitted></i>
<p>* Step 6:</p> <p>Message TransactionEventResponse</p> <ul style="list-style-type: none">- totalCost <i><not omitted></i>
<p>Post scenario validations:</p> <p>N/a</p>

TC_C_126_CSMS: Ad hoc payment via stand-alone payment terminal - local cost calculation

Test case name	Ad hoc payment via stand-alone payment terminal - local cost calculation
Test case Id	TC_C_126_CSMS
Use case Id(s)	C24
Requirement(s)	C24.FR.01, C24.FR.02, C24.FR.10
System under test	CSMS
Description	In order to test that Charging Station supports ad hoc payment via a stand-alone payment terminal with local cost calculation.
Purpose	To verify that the CSMS can properly handle ad hoc payments made via a stand-alone payment terminal when using local cost calculation.
Prerequisite(s)	<ul style="list-style-type: none"> - CSMS is connected with a stand-alone payment terminal - Test System has AuthCtrlr.AuthorizeRemoteStart = <i>false</i> - Test System has TariffCostCtrlr.Enabled[Cost] = <i>true</i> - Test System has TariffCostCtrlr.Currency = "EUR"

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> Present a payment card to the payment terminal Kiosk for Charging Station with identity <Configured unique Charging Station identifier>	
2. The Test System responds with a RequestStartTransactionResponse with status <i>Accepted</i>	1. The CSMS sends a RequestStartTransactionRequest
3. The Test System sends a TransactionEventRequest with eventType <i>Started</i> idToken <IdToken from step 1> idToken.type <i>DirectPayment</i> idToken.additionalInfo <additionalInfo> transactionInfo.transactionId <transactionId>	4. The CSMS responds with a TransactionEventResponse
Settlement	

Main (Test scenario)	
<p>5. The Test System sends a TransactionEventRequest with eventType <i>Ended</i></p> <p>idToken <i><IdToken from step 1></i></p> <p>idToken.type <i>DirectPayment</i></p> <p>transactionInfo.transactionId <i><transactionId></i></p> <p>transactionInfo.transactionLimit.maxCost <i><not omitted></i></p> <p>costDetails.totalCost.currency <i>EUR</i></p> <p>costDetails.totalCost.fixed.exclTax <i>15.00</i></p> <p>costDetails.totalCost.fixed.inclTax <i>18.15 EUR</i></p> <p>costDetails.totalCost.fixed.taxRate.type <i>MyTax</i></p> <p>costDetails.totalCost.fixed.taxRate.tax <i>21</i></p> <p>costDetails.totalUsage.energy <i>123</i></p> <p>costDetails.totalUsage.chargingTime <i>5</i></p> <p>costDetails.totalUsage.idleTime <i>0</i></p> <p>costDetails.chargingPeriods <i><omitted></i></p>	<p>6. The CSMS responds with a TransactionEventResponse</p>
<p><u>Note(s):</u></p> <p>- The CSMS sends a message to Payment Kiosk or PSP to settle the cost for amount: 18.15 EUR</p>	

Tool validations
<p>* Step 1:</p> <p>Message RequestStartTransactionRequest</p> <ul style="list-style-type: none"> - evseld must be <i><evse></i> - remoteStartId must be <i><not omitted></i> - idToken.idToken must be <i><Not empty></i> - idToken.type must be <i>DirectPayment</i> - idToken.additionalInfo.additionalIdToken must be <i><Configured CardLast4Digits></i> - idToken.additionalInfo.type must be <i>CardLast4Digits</i> <p>* Step 4:</p> <p>Message TransactionEventResponse</p> <p>transactionLimit.maxCost = <i><not omitted></i></p>
<p>Post scenario validations:</p> <p>N/a</p>

TC_C_131_CSMS: Ad hoc payment via static or dynamic QR code - success

Test case name	Ad hoc payment via static or dynamic QR code - success
Test case Id	TC_C_131_CSMS
Use case Id(s)	C25
Requirement(s)	C25.FR.07, C25.FR.09, C25.FR.20, C25.FR.21, C25.FR.23, C25.FR.24, C25.FR.40, C25.FR.41
System under test	CSMS
Description	In order to test that CSMS supports QR codes
Purpose	To verify if the CSMS is able to respond correctly when a QR code is scanned on Charging Station
Prerequisite(s)	<ul style="list-style-type: none"> - CSMS configured for central cost calculation - CSMS supports payments using dynamic QR codes - CSMS supports url templates for QR code payments containing at least placeholders {chargingstationid} and {totp} Test System has: <ul style="list-style-type: none"> - WebPaymentsCtrlr.Enabled is <i>true</i> - WebPaymentsCtrlr.URLTemplate is <i><base-url>/{chargingstationid}/{evse}/{totp}/{version}</i> - WebPaymentsCtrlr.ValidityTime is <i>30</i> - WebPaymentsCtrlr.SharedSecret is <i>mysharedsecret</i> - WebPaymentsCtrlr.Length is <i>8</i> - WebPaymentsCtrlr.TOTPVersion is <i>1</i>

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action:</u> Trigger CSMS to perform a payment and start a transaction via QR code webpage maxenergy is 20000 qrUrl is <Configured QR base url> extended with: {chargingstationId} is <configured charging station id> {totp} is <valid totp> {evseld} is <configured evse> {version} is 1	
2. The Test System responds with RequestStartTransactionResponse with status <i>Accepted</i>	1. The CSMS sends a RequestStartTransactionRequest
3. The Test System sends a TransactionEventRequest with eventType <i>Started</i> triggerReason <i>RemoteStart</i> idToken.idToken <i><PspRef></i> idToken.type <i>DirectPayment</i> transactionInfo.transactionId <i><transactionId></i> transactionInfo.remoteStartId <i><remoteStartId></i> meterValue[0].timestamp <i><not omitted></i> meterValue[0].sampledValue[0].value <i>10000</i> meterValue[0].sampledValue[0].context <i>Transaction.Begin</i>	4. The CSMS responds with a TransactionEventResponse

Tool validations
<p>* Step 1:</p> <p>Message RequestStartTransactionRequest</p> <p>idToken.idToken must be <i><PspRef></i></p> <p>idToken.type must be <i>DirectPayment</i></p> <p>* Step 4:</p> <p>Message TransactionEventResponse</p> <p>idTokenInfo.status must be <i>Accepted</i></p> <p>transactionLimit.maxEnergy must be <i>20000</i></p>
<p>Post scenario validations:</p> <p>- N/a</p>

TC_C_132_CSMS: Ad hoc payment via static or dynamic QR code - invalid URL parameters

Test case name	Ad hoc payment via static or dynamic QR code - invalid URL parameters
Test case Id	TC_C_132_CSMS
Use case Id(s)	C25
Requirement(s)	C25.FR.09
System under test	CSMS
Description	In order to test that CSMS supports QR codes
Purpose	To verify if the CSMS is able to respond correctly when receiving invalid parameters.
Prerequisite(s)	<ul style="list-style-type: none"> - CSMS configured for central cost calculation - CSMS supports payments using dynamic QR codes Test System has: <ul style="list-style-type: none"> - WebPaymentsCtrlr.Enabled is <i>true</i> - WebPaymentsCtrlr.URLTemplate is <i><base-url>/{chargingstationid}/{evse}/{totp}/{version}</i> - WebPaymentsCtrlr.ValidityTime is <i>30</i> - WebPaymentsCtrlr.SharedSecret is <i>mysharedsecret</i> - WebPaymentsCtrlr.Length is <i>8</i> - WebPaymentsCtrlr.TOTPVersion is <i>1</i>

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
Manual Action: Trigger CSMS to perform a payment and start a transaction via QR code webpage maxenergy is 20000 qrUrl is <i><Configured QR base url></i> extended with: {chargingstationId} is <i><omitted></i> {totp} is <i><valid totp></i> {evseId} is <i><configured evse></i> {version} is <i>1</i>	
	1. The CSMS does NOT send a RequestStartTransactionRequest

Tool validations
N/a
Post scenario validations: <ul style="list-style-type: none"> - CSMS detects that not all path parameters from URLTemplate are present and shows a web page that QR code is not valid. - CSMS doesn't forward EV Driver to a web page of PSP to enter payment credentials and SHALL NOT authorize EV Driver to charge due missing <i>chargingstationid</i> URL parameter.

TC_C_133_CSMS: Ad hoc payment via static or dynamic QR code - invalid totp

Test case name	Ad hoc payment via static or dynamic QR code - invalid totp
Test case Id	TC_C_133_CSMS
Use case Id(s)	C25
Requirement(s)	C25.FR.07, C25.FR.08
System under test	CSMS
Description	In order to test that Charging Station supports QR codes.
Purpose	To verify if the CSMS is able to respond correctly when receiving invalid TOTP.
Prerequisite(s)	<ul style="list-style-type: none"> - CSMS configured for central cost calculation - CSMS supports payments using dynamic QR codes Test System has: <ul style="list-style-type: none"> - WebPaymentsCtrlr.Enabled is <i>true</i> - WebPaymentsCtrlr.URLTemplate is <i><base-url>/{chargingstationid}/{evse}/{totp}/{version}</i> - WebPaymentsCtrlr.ValidityTime is <i>30</i> - WebPaymentsCtrlr.SharedSecret is <i>mysharedsecret</i> - WebPaymentsCtrlr.Length is <i>8</i> - WebPaymentsCtrlr.TOTPVersion is <i>1</i>

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action:</u> Trigger CSMS to perform a payment and start a transaction via QR code webpage maxenergy is 20000 qrUrl is <Configured QR base url> extended with: {chargingstationId} is <configured charging station id> {totp} is <invalid totp> {evseld} is <configured evse> {version} is 1	
	1. The CSMS does NOT send a RequestStartTransactionRequest

Tool validations
N/a
Post scenario validations: <ul style="list-style-type: none"> - CSMS detects that TOTP is not valid and shows a web page that QR code is not valid. - CSMS doesn't forward EV Driver to a web page of PSP to enter payment credentials and SHALL NOT authorize EV Driver to charge due missing <i>chargingstationid</i> URL parameter.

D Local Authorization List Management

TC_D_01_CSMS: Send Local Authorization List - Full

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2 The Test System responds with a GetLocalListVersionResponse with versionNumber 1	1. The CSMS sends a GetLocalListVersionRequest
<u>Note(s)</u> : This step is optional	
4 The Test System responds with a SendLocalListResponse with status Accepted	3. The CSMS sends a SendLocalListRequest
<u>Note(s)</u> : If the Local Authorization List is too big for one message, step 1 and 2 will be repeated	

Tool validations
<p>* Step 1:</p> <p>Message SendLocalListRequest</p> <ul style="list-style-type: none"> - updateType <i>Full</i> - versionNumber <i><Bigger than 0></i> - localAuthorizationList <i><Not empty></i> - localAuthorizationList[n].idTokenInfo <i><Not empty></i>
<p>Post scenario validations:</p> <ul style="list-style-type: none"> - N/a

TC_D_02_CSMS: Send Local Authorization List - Differential Update

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action:</u> Request the CSMS to send a Local Authorization list to the Charging Station with type Differential and some idTokens in the message	
2 The Test System responds with a GetLocalListVersionResponse with versionNumber 1	1. The CSMS sends a GetLocalListVersionRequest
4 The Test System responds with a SendLocalListResponse with status Accepted	3. The CSMS sends a SendLocalListRequest
<u>Note(s):</u> 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 <i>Differential</i> - versionNumber <i><Bigger than currently configured in Test System></i> - localAuthorizationList <i><Not empty></i>
Post scenario validations: - N/a

TC_D_03_CSMS: Send Local Authorization List - Differential Remove

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action</u> : 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 Test System responds with a SendLocalListResponse with status Accepted	1. The CSMS sends a SendLocalListRequest
<u>Note(s)</u> : 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 <i>Differential</i> - versionNumber <i><Bigger than currently configured in Test System></i> - localAuthorizationList <i><AuthorizationData elements without idTokenInfo></i>
Post scenario validations: - N/a

TC_D_04_CSMS: Send Local Authorization List - Full with empty list

Test case name	Send Local Authorization List - Full with empty 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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action</u> : 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 Test System responds with a SendLocalListResponse with status Accepted	1. The CSMS sends a SendLocalListRequest
<u>Note(s)</u> : 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 <i>Full</i> - localAuthorizationList <i><Empty></i>
Post scenario validations: - N/a

TC_D_08_CSMS: Get Local List Version - Success

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action:</u> Request the CSMS to get a Local Authorization list version	
2 The Test System responds with a GetLocalListVersionResponse with versionNumber <Configured versionNumber>	1. The CSMS sends a GetLocalListVersionRequest

Tool validations
- N/a
Post scenario validations: - N/a

TC_D_09_CSMS: Get Local List Version - No list available

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action:</u> Request the CSMS to get a Local Authorization list version	
2 The Test System responds with a GetLocalListVersionResponse with versionNumber 0	1. The CSMS sends a GetLocalListVersionRequest

Tool validations
- N/a
Post scenario validations: - N/a

E Transactions

TC_E_01_CSMS: Start transaction options - PowerPathClosed

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 Test System 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 Test System notifies the CSMS about the status change of the connector. Message: StatusNotificationRequest - connectorStatus is <i>Occupied</i> Message: NotifyEventRequest - trigger is <i>Delta</i> - actualValue is <i>Occupied</i> - component.name is <i>Connector</i> - variable.name is <i>AvailabilityState</i>	4. The CSMS responds accordingly.
5. The Test System sends a TransactionEventRequest With eventType is <i>Started</i> triggerReason is <i>ChargingStateChanged</i> idToken.idToken <Configured valid_idtoken_idtoken> idToken.type <Configured valid_idtoken_type> evse.id is <Configured evseld> evse.connectorId is <Configured connectorId> transactionInfo.chargingState is <i>SuspendedEVSE</i>	6. The CSMS responds with a TransactionEventResponse
7. The Test System sends a TransactionEventRequest With eventType is <i>Updated</i> triggerReason is <i>ChargingStateChanged</i> transactionInfo.chargingState is <i>Charging</i>	8. The CSMS responds with a TransactionEventResponse

Tool validations
<p>* Step 2:</p> <p>Message: AuthorizeResponse</p> <p>- idTokenInfo.status must be <i>Accepted</i></p> <p>* Step 6:</p> <p>Message: TransactionEventResponse</p> <p>- idTokenInfo.status must be <i>Accepted</i></p>
<p>Post scenario validations:</p> <p>N/a</p>

TC_E_02_CSMS: Start transaction options - EnergyTransfer

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 Test System 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 Test System notifies the CSMS about the status change of the connector. Message: StatusNotificationRequest - connectorStatus is <i>Occupied</i> Message: NotifyEventRequest - trigger is <i>Delta</i> - actualValue is <i>Occupied</i> - component.name is <i>Connector</i> - variable.name is <i>AvailabilityState</i>	4. The CSMS responds accordingly.
5. The Test System sends a TransactionEventRequest With eventType is <i>Started</i> triggerReason is <i>ChargingStateChanged</i> idToken.idToken <Configured valid_idtoken_idtoken> idToken.type <Configured valid_idtoken_type> evse.id is <Configured evseld> evse.connectorId is <Configured connectorId> transactionInfo.chargingState is <i>Charging</i>	6. The CSMS responds with a TransactionEventResponse

Tool validations
* Step 2: Message: AuthorizeResponse - idTokenInfo.status must be <i>Accepted</i> * Step 6: Message: TransactionEventResponse - idTokenInfo.status must be <i>Accepted</i>

Tool validations
Post scenario validations: N/a

TC_E_03_CSMS: Local start transaction - Cable plugin first - Success

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

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>EVConnectedPreSession</i>

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>Authorized</i>	
2. Execute Reusable State <i>EnergyTransferStarted</i>	

Tool validations
N/a
Post scenario validations: N/a

TC_E_04_CSMS: Local start transaction - Authorization first - Success

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

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

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>Authorized</i>	
2. Execute Reusable State <i>EnergyTransferStarted</i>	

Tool validations
N/a
Post scenario validations: N/a

TC_E_39_CSMS: Stop transaction options - Deauthorized - timeout

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 <i>Authorized</i>

Main (Test scenario)	
Charging Station	CSMS
<p>1. The Test System sends a TransactionEventRequest With triggerReason is <i>EVConnectTimeout</i> transactionInfo.stoppedReason is <i>Timeout</i> eventType is <i>Ended</i></p> <p><u>Note(s):</u> - This step will be executed after the <i>_<Configured EV connection timeout></i> expires._</p>	<p>2. The CSMS responds with a TransactionEventResponse</p>

Tool validations
N/a
Post scenario validations: N/a

TC_E_14_CSMS: Stop transaction options - EVDisconnected - Charging Station side

Test case name	Stop transaction options - EVDisconnected - Charging Station side
Test case Id	TC_E_14_CSMS
Use case Id(s)	E06(S2)
Requirement(s)	E06.FR.02
System under test	CSMS
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the stop options.
Purpose	To verify if the CSMS is able to handle a Charging Station that stops a transaction when the EV and EVSE are disconnected at the Charging Station side.
Prerequisite(s)	N/a

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>EVConnectedPostSession</i>

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>EVD</i> Disconnected	

Tool validations
N/a
Post scenario validations: N/a

TC_E_20_CSMS: Stop transaction options - EVDisconnected - EV side (able to charge IEC 61851-1 EV)

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

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferSuspended</i>

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>EVDIsconnected</i>	

Tool validations
N/a
Post scenario validations: N/a

TC_E_15_CSMS: Stop transaction options - StopAuthorized - Local

Test case name	Stop transaction options - StopAuthorized - Local
Test case Id	TC_E_15_CSMS
Use case Id(s)	E06(S3)
Requirement(s)	E06.FR.03
System under test	CSMS
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the stop options.
Purpose	To verify if the CSMS is able to handle a Charging Station that stops a transaction when the EV driver locally stops the transaction.
Prerequisite(s)	N/a

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
1. The Test System sends a TransactionEventRequest With triggerReason is <i>StopAuthorized</i> transactionInfo.stoppedReason is <i>Local</i> eventType is <i>Ended</i>	2. The CSMS responds with a TransactionEventResponse

Tool validations
N/a
Post scenario validations: N/a

TC_E_21_CSMS: Stop transaction options - StopAuthorized - Remote

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 <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action</u> : Trigger the CSMS to request the Charging Station to stop the ongoing transaction.	
2. The Test System responds with a RequestStopTransactionResponse with status <i>Accepted</i>	1. The CSMS sends a RequestStopTransactionRequest
3. The Test System sends a TransactionEventRequest . with triggerReason is <i>RemoteStop</i> transactionInfo.stoppedReason is <i>Remote</i> eventType is <i>Ended</i>	4. The CSMS responds with a TransactionEventResponse .

Tool validations
* Step 1: Message: RequestStopTransactionRequest - transactionId must equal <transactionId provided by the Test System in before state.>
Post scenario validations: N/a

TC_E_09_CSMS: Start transaction options - EVConnected

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
<p>1. The Test System notifies the CSMS about the status change of the connector.</p> <p>Message: StatusNotificationRequest - connectorStatus is <i>Occupied</i> Message: NotifyEventRequest - trigger is <i>Delta</i> - actualValue is <i>Occupied</i> - component.name is <i>Connector</i> - variable.name is <i>AvailabilityState</i></p>	<p>2. The CSMS responds accordingly.</p>
<p>3. The Test System sends a TransactionEventRequest With eventType is <i>Started</i> triggerReason is <i>CablePluggedIn</i> evse.id is <i><Configured evseld></i> evse.connectorId is <i><Configured connectorId></i> transactionInfo.chargingState is <i>EVConnected</i></p>	<p>4. The CSMS responds with a TransactionEventResponse</p>

Tool validations
N/a
Post scenario validations: N/a

TC_E_10_CSMS: Start transaction options - Authorized - Local

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 Test System 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 Test System sends a TransactionEventRequest With eventType is <i>Started</i> triggerReason is <i>Authorized</i> idToken.idToken <Configured valid_idtoken_idtoken> idToken.type <Configured valid_idtoken_type>	4. The CSMS responds with a TransactionEventResponse

Tool validations
<p>* Step 2: Message: AuthorizeResponse - idTokenInfo.status must be <i>Accepted</i></p> <p>* Step 4: Message: TransactionEventResponse - idTokenInfo.status must be <i>Accepted</i></p>
Post scenario validations: N/a

TC_E_11_CSMS: Start transaction options - DataSigned

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 Test System 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 Test System notifies the CSMS about the status change of the connector. Message: StatusNotificationRequest - connectorStatus is <i>Occupied</i> Message: NotifyEventRequest - trigger is <i>Delta</i> - actualValue is <i>Occupied</i> - component.name is <i>Connector</i> - variable.name is <i>AvailabilityState</i>	4. The CSMS responds accordingly.
5. The Test System sends a TransactionEventRequest With eventType is <i>Started</i> triggerReason is <i>SignedDataReceived</i> idToken.idToken <Configured valid_idtoken_idtoken> idToken.type <Configured valid_idtoken_type> evse.id is <Configured evseld> evse.connectorId is <Configured connectorId> meterValue is provided with the following values: sampledValue.value is <i>0.0</i> sampledValue.context is <i>Transaction.Begin</i> sampledValue.signedMeterValue is <Generated <i>SignedMeterValueType</i> >	6. The CSMS responds with a TransactionEventResponse
7. The Test System sends a TransactionEventRequest With eventType is <i>Updated</i> triggerReason is <i>ChargingStateChanged</i> transactionInfo.chargingState is <i>Charging</i>	8. The CSMS responds with a TransactionEventResponse

Tool validations
<p>* Step 2:</p> <p>Message: AuthorizeResponse</p> <p>- idTokenInfo.status must be <i>Accepted</i></p> <p>* Step 6:</p> <p>Message: TransactionEventResponse</p> <p>- idTokenInfo.status must be <i>Accepted</i></p>
<p>Post scenario validations:</p> <p>N/a</p>

TC_E_12_CSMS: Start transaction options - ParkingBayOccupied

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 (Test scenario)	
Charging Station	CSMS
1. The Test System sends a TransactionEventRequest With eventType is <i>Started</i> triggerReason is <i>EVDetected</i>	2. The CSMS responds with a TransactionEventResponse

Tool validations
N/a
Post scenario validations: N/a

TC_E_38_CSMS: Local start transaction - EV not ready

Test case name	Local start transaction - EV not ready
Test case Id	TC_E_38_CSMS
Use case Id(s)	E03
Requirement(s)	N/a
System under test	CSMS
Description	OCPP 2.x.x allows an EV driver to either first connect the EV and EVSE OR present a form of identification. Both sequences will result in being able to charge.
Purpose	To verify if the CSMS is able to handle a Charging Station that reports an EV is not ready to start the energy transfer (yet).
Prerequisite(s)	N/a

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>Authorized</i>

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>EVConnectedPreSession</i>	
2. The Test System sends a TransactionEventRequest With triggerReason is <i>ChargingStateChanged</i> transactionInfo.chargingState is <i>SuspendedEV</i> eventType is <i>Updated</i>	3. The CSMS responds with a TransactionEventResponse

Tool validations
N/a
Post scenario validations: N/a

TC_E_07_CSMS: Stop transaction options - PowerPathClosed - Local stop

Test case name	Stop transaction options - PowerPathClosed - Local stop
Test case Id	TC_E_07_CSMS
Use case Id(s)	E06(S5)
Requirement(s)	E06.FR.06
System under test	CSMS
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the stop options.
Purpose	To verify if the CSMS is able to handle a Charging Station that stops a transaction when it is locally stopped by an EV driver.
Prerequisite(s)	N/a

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
1. The Test System sends a TransactionEventRequest With triggerReason is <i>StopAuthorized</i> transactionInfo.stoppedReason is <i>Local</i> eventType is <i>Ended</i>	2. The CSMS responds with a TransactionEventResponse

Tool validations
N/a
Post scenario validations: N/a

TC_E_08_CSMS: Stop transaction options - EnergyTransfer stopped - StopAuthorized

Test case name	Stop transaction options - EnergyTransfer stopped - StopAuthorized
Test case Id	TC_E_08_CSMS
Use case Id(s)	E06(S6)
Requirement(s)	E06.FR.07
System under test	CSMS
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the stop options.
Purpose	To verify if the CSMS is able to handle a Charging Station that stops a transaction when the energy transfer stopped normally.
Prerequisite(s)	N/a

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>StopAuthorized</i>

Main (Test scenario)	
Charging Station	CSMS
1. The Test System sends a TransactionEventRequest With triggerReason is <i>ChargingStateChanged</i> transactionInfo.chargingState is <i>EVConnected</i> transactionInfo.stoppedReason is <i>Local</i> eventType is <i>Ended</i>	2. The CSMS responds with a TransactionEventResponse

Tool validations
N/a
Post scenario validations: N/a

TC_E_16_CSMS: Stop transaction options - Deauthorized - Invalid idToken

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 (Test scenario)	
Charging Station	CSMS
1. The Test System sends a TransactionEventRequest With triggerReason is <i>Authorized</i> idToken.idToken <Configured invalid_idtoken_idtoken> idToken.type <Configured invalid_idtoken_type> eventType is <i>Started</i>	2. The CSMS responds with a TransactionEventResponse
3. The Test System sends a TransactionEventRequest With eventType <i>Ended</i> triggerReason <i>Deauthorized</i> transactionInfo.stoppedReason <i>DeAuthorized</i>	4. The CSMS responds with a TransactionEventResponse

Tool validations
* Step 2: Message: TransactionEventResponse - idTokenInfo.status must be <i>Invalid</i> or <i>Unknown</i> +
Post scenario validations: N/a

TC_E_17_CSMS: Stop transaction options - Deauthorized - EV side disconnect

Test case name	Stop transaction options - Deauthorized - EV side disconnect
Test case Id	TC_E_17_CSMS
Use case Id(s)	E06(S3)
Requirement(s)	E06.FR.04
System under test	CSMS
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the start options.
Purpose	To verify if the CSMS is able to handle a Charging Station that stops a transaction when the transaction gets deauthorized by a connection loss from the EV side and it has been configured to do so.
Prerequisite(s)	N/a

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferSuspended</i>

Main (Test scenario)	
Charging Station	CSMS
1. The Test System sends a TransactionEventRequest triggerReason must be <i>EVCommunicationLost</i> transactionInfo.chargingState must be <i>Idle</i> transactionInfo.stoppedReason must be <i>EVDisconnected</i> eventType must be <i>Ended</i>	2. The CSMS responds with a TransactionEventResponse

Tool validations
N/a
Post scenario validations: N/a

TC_E_22_CSMS: Stop transaction options - EnergyTransfer stopped - SuspendedEV

Test case name	Stop transaction options - EnergyTransfer stopped - SuspendedEV
Test case Id	TC_E_22_CSMS
Use case Id(s)	E06(S6)
Requirement(s)	E06.FR.07
System under test	CSMS
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the stop options.
Purpose	To verify if the CSMS is able to handle a Charging Station that stops a transaction when the energy transfer stopped by the EV.
Prerequisite(s)	N/a

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
1. The Test System sends a TransactionEventRequest With triggerReason is <i>ChargingStateChanged</i> transactionInfo.chargingState is <i>SuspendedEV</i> transactionInfo.stoppedReason is <i>StoppedByEV</i> eventType is <i>Ended</i>	2. The CSMS responds with a TransactionEventResponse

Tool validations
N/a
Post scenario validations: N/a

TC_E_19_CSMS: Stop transaction options - ParkingBayUnoccupied

Test case name	Stop transaction options - ParkingBayUnoccupied
Test case Id	TC_E_19_CSMS
Use case Id(s)	E06(S1)
Requirement(s)	E06.FR.01
System under test	CSMS
Description	OCPP 2.x.x has a flexible transaction mechanism that allows the transaction start and stop points to be configured differently. This test covers one of the stop options.
Purpose	To verify if the CSMS is able to handle a Charging Station that stops a transaction when the EV left the parking bay.
Prerequisite(s)	N/a

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>EVDisconnected</i>

Main (Test scenario)	
Charging Station	CSMS
1. The Test System sends a TransactionEventRequest With triggerReason is <i>EVDeparted</i> transactionInfo.stoppedReason is <i>Local</i> eventType is <i>Ended</i>	2. The CSMS responds with a TransactionEventResponse

Tool validations
N/a
Post scenario validations: N/a

TC_E_26_CSMS: Disconnect cable on EV-side - Suspend transaction

Test case name	Disconnect cable on EV-side - Suspend transaction
Test case Id	TC_E_26_CSMS
Use case Id(s)	E10
Requirement(s)	E10.FR.01
System under test	CSMS
Description	The Charging Station can behave in several different ways when the cable is disconnected at the EV side, based on its configuration. This test case tests one of the possible configuration settings.
Purpose	To verify if the CSMS can handle a Charging Station that suspends the transaction when the EV and EVSE are disconnected at the EV side AND is able restart the energy transfer after reconnecting the EV and EVSE.
Prerequisite(s)	N/a

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferSuspended</i>

Main (Test scenario)	
Charging Station	CSMS
1. The Test System sends a TransactionEventRequest With triggerReason is <i>EVCommunicationLost</i> transactionInfo.chargingState is <i>Idle</i> eventType is <i>Updated</i>	2. The CSMS responds with a TransactionEventResponse
3. The Test System notifies the CSMS about the current state of the connector. Message: StatusNotificationRequest - connectorStatus <i>Available</i> - evseld <i><Configured evseld></i> - connectorId <i><Configured connectorId></i> Message: NotifyEventRequest - trigger <i>Delta</i> - actualValue <i>"Available"</i> - component.name <i>"Connector"</i> - component.evse.id <i><Configured evseld></i> - component.evse.connectorId <i><Configured connectorId></i> - variable.name <i>"AvailabilityState"</i>	4. The CSMS responds accordingly.
5. The Test System sends a TransactionEventRequest With triggerReason is <i>CablePluggedIn</i> transactionInfo.chargingState is <i>EVConnected</i> eventType is <i>Updated</i>	6. The CSMS responds with a TransactionEventResponse
7. The Test System sends a TransactionEventRequest With triggerReason is <i>ChargingStateChanged</i> transactionInfo.chargingState is <i>Charging</i> eventType is <i>Updated</i>	8. The CSMS responds with a TransactionEventResponse

Tool validations
N/a
Post scenario validations: N/a

TC_E_29_CSMS: Check Transaction status - Transaction with id ongoing - with message in queue

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 Test System will respond that there are message(s) queued belonging to the ongoing transaction with the requested id.
Prerequisite(s)	N/a

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
1. The Test System closes the WebSocket connection.	
2. The Test System waits a number of seconds equal to _<Configured Transaction Duration>, then it will reconnect to the CSMS._	
4. The Test System responds with a GetTransactionStatusResponse With ongoingIndicator is <i>true</i> messagesInQueue is <i>true</i>	3. The CSMS sends a GetTransactionStatusRequest
5. The Test System sends a TransactionEventRequest With eventType is <i>Updated</i> meterValues is present. offline is <i>true</i>	6. The CSMS responds with a TransactionEventResponse

Tool validations
* Step 3: Message: GetTransactionStatusRequest - transactionId <i><Generated transactionId from Before></i>
Post scenario validations: N/a

TC_E_30_CSMS: Check Transaction status - Transaction with id ongoing - without message in queue

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 Test System will respond that there is NO message queued belonging to the ongoing transaction with the requested id.
Prerequisite(s)	N/a

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
2. The Test System responds with a GetTransactionStatusResponse With ongoingIndicator is <i>true</i> messagesInQueue is <i>false</i>	1. The CSMS sends a GetTransactionStatusRequest

Tool validations
* Step 1: Message: GetTransactionStatusRequest - transactionId must be <Generated transactionId from Before>
Post scenario validations: N/a

TC_E_31_CSMS: Check Transaction status - Transaction with id ended - with message in queue

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 Test System will respond that there are message(s) queued belonging to an ended transaction with the requested id.
Prerequisite(s)	N/a

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
1. The Test System closes the WebSocket connection.	
2. The Test System waits a number of seconds equal to <i><Configured Transaction duration></i> , then it will reconnect to the CSMS.	
3. The Test System sends a StatusNotificationRequest With evseld is <i><Configured evseld></i> connectorId is <i><Configured connectorId></i> connectorStatus is <i>Available</i>	4. The CSMS responds with a StatusNotificationResponse
5. The Test System sends a TransactionEventRequest With eventType is <i>Ended</i> offline is <i>true</i> triggerReason is <i>EVCommunicationLost</i> transactionInfo.chargingState is <i>Idle</i> seqNo <i><Skips two sequence number values></i>	6. The CSMS responds with a TransactionEventResponse
8. The Test System responds with a GetTransactionStatusResponse With ongoingIndicator is <i>false</i> messagesInQueue is <i>true</i>	7. The CSMS sends a GetTransactionStatusRequest
9. The Test System sends a TransactionEventRequest With triggerReason is <i>StopAuthorized</i> eventType is <i>Updated</i> offline is <i>true</i> seqNo <i><This is the first of the two skipped values></i>	10. The CSMS responds with a TransactionEventResponse
11. The Test System sends a TransactionEventRequest With triggerReason is <i>ChargingStateChanged</i> transactionInfo.chargingState is <i>EVConnected</i> eventType is <i>Updated</i> offline is <i>true</i> seqNo <i><This is the second of the two skipped values></i>	12. The CSMS responds with a TransactionEventResponse

Tool validations
* Step 5: Message: GetTransactionStatusRequest - transactionId <Generated transactionId from Before>
Post scenario validations: N/a

TC_E_33_CSMS: Check Transaction status - Without transactionId - with message in queue

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 Test System 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 <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
1. The Test System closes the WebSocket connection.	
2. The Test System waits a number of seconds equal to _<Configured Transaction Duration>, then it will reconnect to the CSMS._	
4. The Test System responds with a GetTransactionStatusResponse With ongoingIndicator is omitted. messagesInQueue is <i>true</i>	3. The CSMS sends a GetTransactionStatusRequest
5. The Test System sends a TransactionEventRequest With eventType is <i>Updated</i> meterValues is present. offline is <i>true</i>	6. The CSMS responds with a TransactionEventResponse

Tool validations
* Step 3: Message: GetTransactionStatusRequest - transactionId must be omitted.
Post scenario validations: N/a

TC_E_34_CSMS: Check Transaction status - Without transactionId - without message in queue

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 Test System 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 Test System responds with a GetTransactionStatusResponse With ongoingIndicator is omitted. messagesInQueue is <i>false</i>	1. The CSMS sends a GetTransactionStatusRequest

Tool validations
* Step 1: Message: GetTransactionStatusRequest - transactionId must be omitted.
Post scenario validations: N/a

TC_E_53_CSMS: Reset Sequence Number - CSMS accepting seqNo = 0 at start of transaction

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 <u>Note(s)</u> : New transaction will use seqNo 0 for the first TransactionEventRequest.	
2. Execute Reusable State EVDIsconnected	
3. Execute Reusable State EnergyTransferStarted <u>Note(s)</u> : 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 <i>eventType</i> = Started and <i>seqNo</i> = 0 and answers with a TransactionEventResponse message.
* Step 3: CSMS accepts the message TransactionEventRequest with <i>eventType</i> = Started and <i>seqNo</i> = 0 and answers with a TransactionEventResponse message.
Post scenario validations: N/a

TC_E_102_CSMS: Transactions with fixed cost, energy or time - CSMS and CS both specify limits

Test case name	Transactions with fixed cost, energy or time - CSMS and CS both specify limits
Test case Id	TC_E_102_CSMS
Use case Id(s)	E16
Requirement(s)	E16.FR.01, E16.FR.02, E16.FR.03
System under test	CSMS
Description	EV Driver or CSMS specifies a limit in cost, energy, state of charge or time for transaction.
Purpose	To verify whether the CSMS correctly handles transactions where both the CSMS and Charging Station specify limits.
Prerequisite(s)	- CSMS allows specifying an energy limit. - Test System has TxCtrlr.SupportedLimits contains <i>maxEnergy</i> .

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
1. The Test System sends a TransactionEventRequest with eventType <i>Updated</i> triggerReason <i>LimitSet</i> transactionInfo.transactionLimit.maxEnergy <i>6000</i>	2. The CSMS responds with a TransactionEventResponse
<u>Manual action:</u> Configure the CSMS to override the max energy limit to 10000 (Wh).	
3. The Test System sends a TransactionEventRequest with eventType <i>Updated</i>	4. The CSMS responds with a TransactionEventResponse
5. The Test System sends a TransactionEventRequest with eventType <i>Updated</i> triggerReason <i>LimitSet</i> transactionInfo.transactionLimit.maxEnergy is <i>10000</i>	6. The CSMS responds with a TransactionEventResponse
7. The Test System sends a TransactionEventRequest with eventType <i>Ended</i> triggerReason <i>EnergyLimitReached</i>	8. The CSMS responds with a TransactionEventResponse

Tool validations
* Step 4: Message: TransactionEventResponse - transactionLimit.maxEnergy must be <i>10000</i>
* Step 6: Message: TransactionEventResponse - transactionLimit is <i><omitted></i>
Post scenario validations: N/a

TC_E_106_CSMS: Transactions with fixed cost, energy or time - CS specifies energy limit

Test case name	Transactions with fixed cost, energy or time - CS specifies energy limit
Test case Id	TC_E_106_CSMS
Use case Id(s)	E16
Requirement(s)	...
System under test	CSMS
Description	EV Driver or CSMS specifies a limit in cost, energy, state of charge or time for transaction.
Purpose	To verify whether the CSMS correctly handles transactions where the Charging Station specifies an energy limit.
Prerequisite(s)	Test System has TxCtrlr.SupportedLimits contains <i>maxEnergy</i> .

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
1. The Test System sends a TransactionEventRequest with eventType <i>Updated</i> triggerReason <i>LimitSet</i> transactionInfo.transactionLimit.maxEnergy <i>6000</i>	2. The CSMS responds with a TransactionEventResponse
3. The Test System sends a TransactionEventRequest with eventType <i>Ended</i> triggerReason <i>EnergyLimitReached</i>	4. The CSMS responds with a TransactionEventResponse

Tool validations
* Step 2: Message TransactionEventResponse with: * transactionLimit is <i><omitted></i>
Post scenario validations: N/a

TC_E_107_CSMS: Transactions with fixed cost, energy or time - CS specifies time limit

Test case name	Transactions with fixed cost, energy or time - CS specifies time limit
Test case Id	TC_E_107_CSMS
Use case Id(s)	E16
Requirement(s)	...
System under test	CSMS
Description	EV Driver or CSMS specifies a limit in cost, energy, state of charge or time for transaction.
Purpose	To verify whether the CSMS correctly handles transactions where the Charging Station specifies a time limit.
Prerequisite(s)	Test System has TxCtrl.SupportedLimits contains <i>maxTime</i> .

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
1. The Test System sends a TransactionEventRequest with eventType <i>Updated</i> triggerReason <i>LimitSet</i> transactionInfo.transactionLimit.maxTime 120	2. The CSMS responds with a TransactionEventResponse
<u>Note</u> : The test system waits for the <i>maxTime transactionLimit</i> to be reached.	
3. The Test System sends a TransactionEventRequest with eventType <i>Ended</i> triggerReason <i>TimeLimitReached</i>	4. The CSMS responds with a TransactionEventResponse

Tool validations
* Step 2: Message TransactionEventResponse with: transactionLimit is <omitted>
Post scenario validations: N/a

TC_E_108_CSMS: Transactions with fixed cost, energy or time - CS calculates costs and specifies limit

Test case name	Transactions with fixed cost, energy or time - CS calculates costs and specifies limit
Test case Id	TC_E_108_CSMS
Use case Id(s)	E16
Requirement(s)	...
System under test	CSMS
Description	EV Driver or CSMS specifies a limit in cost, energy, state of charge or time for transaction.
Purpose	To verify whether the CSMS correctly handles transactions where the Charging Station uses local cost calculation and specifies a cost limit.
Prerequisite(s)	<ul style="list-style-type: none"> - Test System has TariffCostCtrlr.Enabled[Cost] is <i>true</i> - Test System has TariffCostCtrlr.Enabled[RunningCost] is <i>true</i>

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 CSMS to configure a tariff with 1 EUR / minute for charging time	
2. The Test System responds with a SetDefaultTariffResponse	1. The CSMS sends a SetDefaultTariffRequest with evseld 0 tariff.tariffId <not omitted> tariff.currency EUR tariff.chargingTime.prices[0].priceMinute 1.00
3. Execute Reusable State <i>EnergyTransferStarted</i>	
4. The Test System sends a TransactionEventRequest with eventType Updated triggerReason LimitSet transactionInfo.transactionLimit.maxCost 2.00 costDetails.totalCost.currency EUR costDetails.totalCost.typeOfCost NormalCost costDetails.totalCost.chargingTime.inclTax 1.00 costDetails.totalCost.total.inclTax 1.00 costDetails.totalUsage.chargingTime 60 costDetails.totalUsage.idleTime <not omitted>	5. The CSMS responds with a TransactionEventResponse
6. The Test System sends a TransactionEventRequest with eventType Updated triggerReason RunningCost costDetails.totalCost.currency EUR costDetails.totalCost.typeOfCost NormalCost costDetails.totalCost.chargingTime.inclTax 1.50 costDetails.totalCost.total.inclTax 1.50 costDetails.totalUsage.chargingTime 90 costDetails.totalUsage.idleTime <not omitted>	7. The CSMS responds with a TransactionEventResponse

Main (Test scenario)	
<p>8. The Test System sends a TransactionEventRequest with eventType <i>Ended</i> triggerReason <i>CostLimitReached</i> costDetails.totalCost.currency <i>EUR</i> costDetails.totalCost.typeOfCost <i>NormalCost</i> costDetails.totalCost.chargingTime.inclTax <i>2.00</i> costDetails.totalCost.total.inclTax <i>2.00</i> costDetails.totalUsage.chargingTime <i>120</i> costDetails.totalUsage.idleTime <i><not omitted></i></p>	<p>9. The CSMS responds with a TransactionEventResponse</p>
Tool validations	
<p>* Step 1: Message: SetDefaultTariffRequest - evseld must be <i><evse id></i> - tariff.tariffId must be <i><not omitted></i> - tariff.currency must be <i>EUR</i> - tariff.chargingTime.prices[0].priceMinute must be <i>1.00</i></p> <p>* Step 5: Message: TransactionEventResponse - totalCost must be <i><omitted></i> - transactionLimit must be <i><omitted></i></p> <p>* Step 7: Message: TransactionEventResponse - totalCost must be <i><omitted></i> - transactionLimit must be <i><omitted></i></p> <p>* Step 9: Message: TransactionEventResponse - totalCost must be <i><omitted></i> - transactionLimit must be <i><omitted></i></p>	
<p>Post scenario validations: N/a</p>	

TC_E_109_CSMS: Transactions with fixed cost, energy or time - CSMS calculates costs and specifies cost limit

Test case name	Transactions with fixed cost, energy or time - CSMS calculates costs and specifies cost limit
Test case Id	TC_E_109_CSMS
Use case Id(s)	E16
Requirement(s)	E16.FR.02, E16.FR.11
System under test	CSMS
Description	CSMS will set a limit the transaction for the specified cost. CS will use central cost calculation.
Purpose	To verify whether the CSMS correctly sends cost when central cost calculation is used.
Prerequisite(s)	<ul style="list-style-type: none"> - CSMS supports setting cost limits - Test System has TxCtrlr.SupportedLimits containing <i>maxCost</i> - Test System has TariffCostCtrlr.Enabled[Cost] is <i>false</i>

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>EVConnectedPreSession</i>

Main (Test scenario)	
Charging Station	CSMS
1. The Test System sends a AuthorizeRequest With idToken.idToken <Configured valid_idtoken_idtoken> idToken.type <Configured valid_idtoken_type>	2. The CSMS responds with a AuthorizeResponse
3. The Test System sends a TransactionEventRequest with With eventType is <i>Started</i> triggerReason is <i>ChargingStateChanged</i> idToken.idToken <Configured valid_idtoken_idtoken> idToken.type <Configured valid_idtoken_type> evse.id is <Configured evseld> evse.connectorId is <Configured connectorId> transactionInfo.transactionId is <transactionId> transactionInfo.chargingState is <i>Charging</i>	4. The CSMS responds with a TransactionEventResponse
5 The Test System sends a TransactionEventRequest with triggerReason is <i>LimitSet</i> transactionInfo.transactionLimit.maxCost is 10	6 The CSMS responds with TransactionEventResponse
8. The Test System responds with a CostUpdatedResponse	7. The CSMS sends a CostUpdatedRequest
<u>Note:</u> Sending a CostUpdatedRequest is not required, but if CSMS does send it, the validation applies.	

Tool validations
<p>* Step 2:</p> <p>Message: AuthorizeResponse</p> <p>idTokenInfo.status must be <i>Accepted</i></p>
<p>* Step 4:</p> <p>Message: TransactionEventResponse</p> <p>totalCost must be <i><not omitted></i></p> <p>transactionInfo.transactionLimit.maxEnergy must be <i><omitted></i></p> <p>transactionInfo.transactionLimit.maxTime must be <i><omitted></i></p> <p>transactionInfo.transactionLimit.maxCost must be <i>10</i></p>
<p>* Step 6:</p> <p>Message: TransactionEventResponse</p> <p>totalCost must be <i><not omitted></i></p> <p>transactionInfo.transactionLimit must be <i><omitted></i></p>
<p>* Step 8:</p> <p>Message: CostUpdatedRequest</p> <p>transactionId must be <i><transactionId></i></p> <p>totalCost must be <i><not omitted></i></p>
<p>Post scenario validations:</p> <p>N/a</p>

TC_E_110_CSMS: Transactions with fixed cost, energy or time - CSMS specifies energy limit

Test case name	Transactions with fixed cost, energy or time - CSMS specifies energy limit
Test case Id	TC_E_110_CSMS
Use case Id(s)	E16
Requirement(s)	E16.FR.02
System under test	CSMS
Description	CSMS will set an energy limit on the transaction.
Purpose	To verify whether the CSMS is able to set an energy limit.
Prerequisite(s)	- CSMS supports setting energy limits - Test System has TxCtrlr.SupportedLimits containing <i>maxEnergy</i>

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>EVConnectedPreSession</i>

Main (Test scenario)	
Charging Station	CSMS
1. The Test System sends a TransactionEventRequest with With eventType is <i>Started</i> triggerReason is <i>ChargingStateChanged</i> idToken.idToken <Configured <i>valid_idtoken_idtoken</i> > idToken.type <Configured <i>valid_idtoken_type</i> > evse.id is <Configured <i>evseld</i> > evse.connectorId is <Configured <i>connectorId</i> > transactionInfo.transactionId is < <i>transactionId</i> > transactionInfo.chargingState is <i>Charging</i> transactionInfo.transactionLimit < <i>omitted</i> >	2. The CSMS responds with a TransactionEventResponse
3 The Test System sends a TransactionEventRequest with triggerReason is <i>LimitSet</i> transactionInfo.transactionLimit.maxEnergy is <Configured <i>energy limit</i> >	4 The CSMS responds with TransactionEventResponse
5. The Test System sends a TransactionEventRequest with eventType is <i>Updated</i> triggerReason is <i>EnergyLimitReached</i> evse.id is <Configured <i>evseld</i> > evse.connectorId is <Configured <i>connectorId</i> > transactionInfo.transactionId is < <i>transactionId</i> > transactionInfo.transactionLimit.maxEnergy <Configured <i>energy limit</i> > transactionInfo.transactionLimit.maxTime < <i>omitted</i> > transactionInfo.transactionLimit.maxCost < <i>omitted</i> > transactionInfo.chargingState <i>SuspendedEVSE</i>	6. The CSMS responds with a TransactionEventResponse

Tool validations
<p>* Step 2:</p> <p>Message: TransactionEventResponse</p> <p>idTokenInfo.status must be <i>Accepted</i></p> <p>transactionLimit.maxEnergy must be <i><Configured energy limit></i></p> <p>transactionLimit.maxTime must be <i><omitted></i></p> <p>transactionLimit.maxCost must be <i><omitted></i></p>
<p>* Step 4:</p> <p>Message: TransactionEventResponse</p> <p>transactionLimit must be <i><omitted></i></p>
<p>Post scenario validations:</p> <p>N/a</p>

TC_E_111_CSMS: Transactions with fixed cost, energy or time - CSMS specifies time limit

Test case name	Transactions with fixed cost, energy or time - CSMS specifies time limit
Test case Id	TC_E_111_CSMS
Use case Id(s)	E16
Requirement(s)	E16.FR.02
System under test	CSMS
Description	CSMS will set a time limit on the transaction.
Purpose	To verify whether the CSMS is able to set a time limit.
Prerequisite(s)	- CSMS supports setting energy limits - Test System has TxCtrlr.SupportedLimits containing <i>maxTime</i>

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>EVConnectedPreSession</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Manual action:</u> In CSMS, configure time limit of duration <Configured time limit> for this transaction	
<p>1. The Test System sends a TransactionEventRequest with</p> <p>With eventType is <i>Started</i></p> <p>triggerReason is <i>ChargingStateChanged</i></p> <p>idToken.idToken <Configured valid_idtoken_idtoken></p> <p>idToken.type <Configured valid_idtoken_type></p> <p>evse.id is <Configured evseld></p> <p>evse.connectorId is <Configured connectorId></p> <p>transactionInfo.transactionId is <transactionId></p> <p>transactionInfo.chargingState is <i>Charging</i></p> <p>transactionInfo.transactionLimit <omitted></p>	<p>2. The CSMS responds with a TransactionEventResponse</p>
<p>3. The Test System sends a TransactionEventRequest with</p> <p>eventType is <i>Updated</i></p> <p>triggerReason is <i>LimitSet</i></p> <p>evse.id is <Configured evseld></p> <p>evse.connectorId is <Configured connectorId></p> <p>transactionInfo.transactionId is <transactionId></p> <p>transactionInfo.transactionLimit.maxEnergy <omitted></p> <p>transactionInfo.transactionLimit.maxTime <Configured time limit></p> <p>transactionInfo.transactionLimit.maxCost <omitted></p>	<p>4. The CSMS responds with a TransactionEventResponse</p>

Tool validations
<p>* Step 2:</p> <p>Message: TransactionEventResponse</p> <p>idTokenInfo.status must be <i>Accepted</i></p> <p>transactionLimit.maxEnergy must be <i><omitted></i></p> <p>transactionLimit.maxTime must be <i><Configured time limit></i></p> <p>transactionLimit.maxCost must be <i><omitted></i></p>
<p>* Step 4:</p> <p>Message: TransactionEventResponse</p> <p>transactionLimit must be <i><omitted></i></p>
<p>Post scenario validations:</p> <p>N/a</p>

TC_E_113_CSMS: Resuming transaction after interruption - TxResumptionTimeout not expired - TxAllowEnergyTransferResumption is true

Test case name	Resuming transaction after interruption - TxResumptionTimeout not expired - TxAllowEnergyTransferResumption is true
Test case Id	TC_E_113_CSMS
Use case Id(s)	E17
Requirement(s)	E17.FR.15
System under test	CSMS
Description	CSMS restores TxProfile after Charging Station resumes transactions which had a Charging profile with purpose TxProfile configured.
Purpose	To verify the CSMS restores the charging profile for a transaction after resuming transactions by the Charging Station.
Prerequisite(s)	N/a

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
The Test System disconnects the connection.	
The Test System waits the <Configured transaction duration>	
The Test System restores the connection.	
1. The Test System sends a BootNotificationRequest	2. The CSMS responds with a BootNotificationResponse
3. The Test System notifies the CSMS about the current state of all connectors. Message: NotifyEventRequest - eventData[0].trigger <i>Delta</i> - eventData[0].component.name <i>"Connector"</i> - eventData[0].variable.name <i>"AvailabilityState"</i> Configured EVSE.Connector: - eventData[0].actualValue <i>"Occupied"</i> Other connectors: - eventData[0].actualValue <i>"Available"</i>	4. The CSMS responds accordingly.
5 The Test System sends a SecurityEventNotificationRequest - type must be <i>StartupOfTheDevice</i> OR <i>ResetOrReboot</i>	6 The CSMS responds with a SecurityEventNotificationResponse
7 The Test System sends a NotifyEventRequest eventData.trigger = <i>Delta</i> eventData.transactionId = <i><transactionId from this transaction></i> eventData.eventNotificationType = <i>HardWiredNotification</i> eventData.severity = 3 eventData.actualValue = <i>true</i> eventData.component.name = <i>ElectricalFeed</i> eventData.variable.name = <i>Problem</i>	8 The CSMS responds with a NotifyEventResponse
9 The Test System sends a NotifyEventRequest Same content as step 7, but with actualValue = <i>false</i> : eventData.actualValue = <i>false</i>	10 The CSMS responds with a NotifyEventResponse

Main (Test scenario)	
11. The Test System sends a TransactionEventRequest eventType <i>Updated</i> triggerReason <i>TxResumed</i> transactionInfo.transactionId <i><transactionId></i> transactionInfo.chargingState <i>Charging</i>	12. The CSMS responds with a TransactionEventResponse
Tool validations	
* Step 2: Message: BootNotificationResponse - status <i>Accepted</i>	
Post scenario validations: N/a	

TC_E_117_CSMS: Set Charging Profile - Resuming transaction after interruption - TxResumptionTimeout not expired - TxAllowEnergyTransferResumption is true

Test case name	Set Charging Profile - Resuming transaction after interruption - TxResumptionTimeout not expired - TxAllowEnergyTransferResumption is true
Test case Id	TC_E_117_CSMS
Use case Id(s)	E17
Requirement(s)	E17.FR.15
System under test	CSMS
Description	CSMS restores TxProfile after Charging Station resumes transactions which had a Charging profile with purpose TxProfile configured.
Purpose	To verify the CSMS restores the charging profile for a transaction after resuming transactions by the Charging Station.
Prerequisite(s)	N/a

Before (Preparations)
Configuration State: SmartChargingCtrlr.ChargingProfilePersistence is <absent>
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Manual action:</u> trigger CSMS to set a Charging Profile of type TxProfile on <transactionId>	
2. The Test System responds with a SetChargingProfileResponse with status <i>Accepted</i>	1. The CSMS sends a SetChargingProfileRequest
<i>The Test System disconnects the connection.</i>	
<i>The Test System waits the <Configured transaction duration></i>	
<i>The Test System restores the connection.</i>	
3. The Test System sends a BootNotificationRequest	4. The CSMS responds with a BootNotificationResponse
5. The Test System notifies the CSMS about the current state of all connectors. Message: NotifyEventRequest - eventData[0].trigger <i>Delta</i> - eventData[0].component.name <i>"Connector"</i> - eventData[0].variable.name <i>"AvailabilityState"</i> Configured EVSE.Connector: - eventData[0].actualValue <i>"Occupied"</i> Other connectors: - eventData[0].actualValue <i>"Available"</i>	6. The CSMS responds accordingly.
7 The Test System sends a SecurityEventNotificationRequest - type must be <i>StartupOfTheDevice</i> OR <i>ResetOrReboot</i>	8 The CSMS responds with a SecurityEventNotificationResponse

Main (Test scenario)	
9 The Test System sends a NotifyEventRequest eventData.trigger = <i>Delta</i> eventData.transactionId = <i><transactionId from this transaction></i> eventData.eventNotificationType = <i>HardWiredNotification</i> eventData.severity = 3 eventData.actualValue = <i>true</i> eventData.component.name = <i>ElectricalFeed</i> eventData.variable.name = <i>Problem</i>	10 The CSMS responds with a NotifyEventResponse
11 The Test System sends a NotifyEventRequest Same content as step 9, but with actualValue = <i>false</i> : eventData.actualValue = <i>false</i>	12 The CSMS responds with a NotifyEventResponse
13. The Test System sends a TransactionEventRequest eventType <i>Updated</i> triggerReason <i>TxResumed</i> transactionInfo.transactionId <i><transactionId></i> transactionInfo.chargingState <i>Charging</i>	14. The CSMS responds with a TransactionEventResponse
16. The Test System responds with a SetChargingProfileResponse with status <i>Accepted</i>	15. The CSMS sends a SetChargingProfileRequest

Tool validations
* Step 1: Message SetChargingProfileRequest - chargingProfile.chargingProfilePurpose must be <i>TxProfile</i> * Step 4: Message: BootNotificationResponse - status must be <i>Accepted</i> * Step 15: Message: SetChargingProfileRequest must contain an identical <i>chargingProfile</i> as in Step 1 or identical as of time of Step 15.
Post scenario validations: N/a

F Remote Control

TC_F_01_CSMS: Remote start transaction - Cable plugin first

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 <i>EVConnectedPreSession</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action:</u> <i>Trigger the CSMS to request the Charging Station to start a transaction.</i>	
2. The Test System responds with a RequestStartTransactionResponse with status <i>Accepted</i> transactionId is <i><Generated transactionId></i>	1. The CSMS sends a RequestStartTransactionRequest
3. The Test System sends a TransactionEventRequest . with triggerReason is <i>RemoteStart</i> transactionInfo.remoteStartId is <i><By CSMS provided remoteStartId></i> eventType is <i>Updated</i>	4. The CSMS responds with a TransactionEventResponse .
5. Execute Reusable State <i>EnergyTransferStarted</i> (State is <i>Authorized</i> and <i>_EVConnected</i> = true)	

Tool validations
* Step 1: Message: RequestStartTransactionRequest - idToken.idToken <i><Configured valid_idtoken_idtoken></i> - idToken.type <i><Configured valid_idtoken_type></i>
Post scenario validations: N/a

TC_F_02_CSMS: Remote start transaction - Remote start first - AuthorizeRemoteStart is true

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 Test System responds with a RequestStartTransactionResponse with status <i>Accepted</i> transactionId is omitted.	1. The CSMS sends a RequestStartTransactionRequest
3. The Test System 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 Test System sends a TransactionEventRequest . with triggerReason is <i>RemoteStart</i> transactionInfo.remoteStartId is <By Test System generated remoteStartID> eventType is <i>Started</i>	6. The CSMS responds with a TransactionEventResponse .
7. Execute Reusable State <i>EnergyTransferStarted</i> (State is <i>Authorized</i> and <i>_EVConnected</i> = false)	

Tool validations
<p>* Step 1: Message: RequestStartTransactionRequest - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type></p> <p>* Step 4: Message: AuthorizeResponse - idTokenInfo.status must be <i>Accepted</i></p>
Post scenario validations: N/a

TC_F_03_CSMS: Remote start transaction - Remote start first - AuthorizeRemoteStart is false

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 Test System responds with a RequestStartTransactionResponse with status <i>Accepted</i> and transactionId is omitted.	1. The CSMS sends a RequestStartTransactionRequest
3. The Test System sends a TransactionEventRequest with triggerReason is <i>RemoteStart</i> and transactionInfo.remoteStartId is <i><By Test System generated remoteStartId></i> and eventType is <i>Started</i>	4. The CSMS responds with a TransactionEventResponse .
5. Execute Reusable State <i>EnergyTransferStarted</i> (State is <i>Authorized</i> and <i>_EVConnected</i> = false)	

Tool validations

* Step 1:

Message: **RequestStartTransactionRequest**- **idToken.idToken** *<Configured valid_idtoken_idtoken>*- **idToken.type** *<Configured valid_idtoken_type>*

Post scenario validations:

N/a

TC_F_04_CSMS: Remote start transaction - Remote start first - Cable plugin timeout

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 Test System responds with a RequestStartTransactionResponse with status <i>Accepted</i> transactionId is omitted.	1. The CSMS sends a RequestStartTransactionRequest
3. The Test System sends a TransactionEventRequest . with triggerReason is <i>RemoteStart</i> transactionInfo.remoteStartId is <i><By Test System generated remoteStartID></i> eventType is <i>Started</i>	4. The CSMS responds with a TransactionEventResponse .
5. The Test System sends a TransactionEventRequest . with triggerReason is <i>EVConnectTimeout</i> eventType is <i>Updated</i>	6. The CSMS responds with a TransactionEventResponse .
<u>Note(s):</u> - This step will be executed after the <i><Configured Transaction Duration></i> has been reached._	

Tool validations
* Step 1: Message: RequestStartTransactionRequest - idToken.idToken <i><Configured valid_idtoken_idtoken></i> - idToken.type <i><Configured valid_idtoken_type></i>
Post scenario validations: N/a

TC_F_06_CSMS: Remote unlock Connector - Without ongoing transaction - Accepted

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 send 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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Test System responds with a UnlockConnectorResponse with status <i>Unlocked</i>	1. The CSMS sends a UnlockConnectorRequest

Tool validations
* Step 1: Message UnlockConnectorRequest - evseld <Configured evseld> - connectorId <Configured connectorId>
Post scenario validations: - N/a

TC_F_11_CSMS: Trigger message - MeterValues - Specific EVSE

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 Test System responds with a TriggerMessageResponse with status <i>Accepted</i>	1. The CSMS sends a TriggerMessageRequest
3. The Test System sends a MeterValuesRequest With evseld <Configured evseld> meterValue[0].sampledValue.context <i>Trigger</i>	4. The CSMS responds with a MeterValuesResponse

Tool validations
* Step 1: Message: TriggerMessageRequest - requestedMessage must be <i>MeterValues</i> - evse.id must be <Configured evseld>
Post scenario validations: N/a

TC_F_12_CSMS: Trigger message - MeterValues - All EVSE

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 Test System responds with a TriggerMessageResponse with status <i>Accepted</i>	1. The CSMS sends a TriggerMessageRequest
3. The Test System sends a MeterValuesRequest With evseld omitted meterValue[0].sampledValue.context <i>Trigger</i>	4. The CSMS responds with a MeterValuesResponse
<u>Note(s):</u> - This step will be executed for every EVSE.	

Tool validations
* Step 1: Message: TriggerMessageRequest - requestedMessage must be <i>MeterValues</i>
Post scenario validations: N/a

TC_F_13_CSMS: Trigger message - TransactionEvent - Specific EVSE

Test case name	Trigger message - TransactionEvent - Specific EVSE
Test case Id	TC_F_13_CSMS
Use case Id(s)	F06
Requirement(s)	F06.FR.01,F06.FR.02
System under test	CSMS
Description	The CSMS can request a Charging Station to send Charging Station-initiated messages. In the request the CSMS indicates which message it wishes to receive.
Purpose	To verify if the CSMS is able to trigger the Charging Station to send a TransactionEventRequest for a specific EVSE, using a TriggerMessageRequest.
Prerequisite(s)	N/a

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
2. The Test System responds with a TriggerMessageResponse with status <i>Accepted</i>	1. The CSMS sends a TriggerMessageRequest
3. The Test System sends a TransactionEventRequest With evse.id <Configured evseld> triggerReason <i>Trigger</i> transactionInfo.chargingState <i>Charging</i> meterValue is present meterValue[0].sampledValue.context <i>Trigger</i>	4. The CSMS responds with a TransactionEventResponse

Tool validations
* Step 1: Message: TriggerMessageRequest - requestedMessage must be <i>TransactionEvent</i> - evse.id must be <Configured evseld>
Post scenario validations: N/a

TC_F_14_CSMS: Trigger message - TransactionEvent - All EVSE

Test case name	Trigger message - TransactionEvent - All EVSE
Test case Id	TC_F_14_CSMS
Use case Id(s)	F06
Requirement(s)	F06.FR.01
System under test	CSMS
Description	The CSMS can request a Charging Station to send Charging Station-initiated messages. In the request the CSMS indicates which message it wishes to receive.
Purpose	To verify if the CSMS is able to trigger the Charging Station to send a TransactionEventRequest for all EVSE, using a TriggerMessageRequest.
Prerequisite(s)	N/a

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
2. The Test System responds with a TriggerMessageResponse with status <i>Accepted</i>	1. The CSMS sends a TriggerMessageRequest
3. The Test System sends a TransactionEventRequest With evse.id omitted triggerReason <i>Trigger</i> transactionInfo.chargingState <i>Charging</i> meterValue is present meterValue[0].sampledValue.context <i>Trigger</i> <u>Note(s):</u> - <i>This step will be executed for every EVSE.</i>	4. The CSMS responds with a TransactionEventResponse

Tool validations
* Step 1: Message: TriggerMessageRequest - requestedMessage must be <i>TransactionEvent</i>
Post scenario validations: N/a

TC_F_15_CSMS: Trigger message - LogStatusNotification - Idle

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 Test System responds with a TriggerMessageResponse with status <i>Accepted</i>	1. The CSMS sends a TriggerMessageRequest
3. The Test System sends a LogStatusNotificationRequest with status <i>Idle</i>	4. The CSMS responds with a LogStatusNotificationResponse

Tool validations
* Step 1: Message: TriggerMessageRequest - requestedMessage must be <i>LogStatusNotification</i>
Post scenario validations: N/a

TC_F_18_CSMS: Trigger message - FirmwareStatusNotification - Idle

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 Test System responds with a TriggerMessageResponse with status <i>Accepted</i>	1. The CSMS sends a TriggerMessageRequest
3. The Test System sends a FirmwareStatusNotificationRequest with status <i>Idle</i>	4. The CSMS responds with a FirmwareStatusNotificationResponse

Tool validations
* Step 1: Message: TriggerMessageRequest - requestedMessage must be <i>FirmwareStatusNotification</i>
Post scenario validations: N/a

TC_F_20_CSMS: Trigger message - Heartbeat

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 Test System responds with a TriggerMessageResponse with status <i>Accepted</i>	1. The CSMS sends a TriggerMessageRequest
3. The Test System sends a HeartbeatRequest	4. The CSMS responds with a HeartbeatResponse

Tool validations
* Step 1: Message: TriggerMessageRequest - requestedMessage must be <i>Heartbeat</i>
Post scenario validations: N/a

TC_F_23_CSMS: Trigger message - StatusNotification - Specific EVSE - Available

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 Test System responds with a TriggerMessageResponse with status <i>Accepted</i>	1. The CSMS sends a TriggerMessageRequest
3. The Test System notifies the CSMS about the current state of the connector. Message: StatusNotificationRequest - connectorStatus <i>Available</i> - evseld <i><Configured evseld></i> - connectorId <i><Configured connectorId></i> Message: NotifyEventRequest - trigger <i>Delta</i> - actualValue <i>"Available"</i> - component.name <i>"Connector"</i> - component.evse.id <i><Configured evseld></i> - component.evse.connectorId <i><Configured connectorId></i> - variable.name <i>"AvailabilityState"</i>	4. The CSMS responds accordingly.

Tool validations
* Step 1: Message: TriggerMessageRequest - requestedMessage must be <i>StatusNotification</i> - evse.id must be <i><Configured evseld></i>
Post scenario validations: N/a

TC_F_24_CSMS: Trigger message - StatusNotification - Specific EVSE - Occupied

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
<p>1. The Test System notifies the CSMS about the current state of the connector.</p> <p>Message: StatusNotificationRequest</p> <ul style="list-style-type: none"> - connectorStatus <i>Occupied</i> - evseld <Configured evseld> - connectorId <Configured connectorId> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none"> - trigger <i>Delta</i> - actualValue <i>"Occupied"</i> - component.name <i>"Connector"</i> - component.evse.id <Configured evseld> - component.evse.connectorId <Configured connectorId> - variable.name <i>"AvailabilityState"</i> 	<p>2. The CSMS responds accordingly.</p>
<p>4. The Test System responds with a TriggerMessageResponse with status <i>Accepted</i></p>	<p>3. The CSMS sends a TriggerMessageRequest</p>
<p>5. The Test System notifies the CSMS about the current state of the connector.</p> <p>Message: StatusNotificationRequest</p> <ul style="list-style-type: none"> - connectorStatus <i>Occupied</i> - evseld <Configured evseld> - connectorId <Configured connectorId> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none"> - trigger <i>Delta</i> - actualValue <i>"Occupied"</i> - component.name <i>"Connector"</i> - component.evse.id <Configured evseld> - component.evse.connectorId <Configured connectorId> - variable.name <i>"AvailabilityState"</i> 	<p>6. The CSMS responds accordingly.</p>

Tool validations
<p>* Step 1:</p> <p>Message: TriggerMessageRequest</p> <ul style="list-style-type: none">- requestedMessage must be <i>StatusNotification</i>- evse.id must be <i><Configured evseld></i>
<p>Post scenario validations:</p> <p>N/a</p>

TC_F_27_CSMS: Trigger message - NotImplemented

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 Test System responds with a TriggerMessageResponse with status <i>NotImplemented</i>	1. The CSMS sends a TriggerMessageRequest

Tool validations
N/a
Post scenario validations: N/a

TC_F_100_CSMS: Trigger message - CustomTrigger

Test case name	Trigger message - CustomTrigger
Test case Id	TC_F_100_CSMS
Use case Id(s)	F06
Requirement(s)	...
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 messages using a CustomTrigger.
Prerequisite(s)	The Test System reports support for the following: component.name <i>CustomizationCtrlr</i> variable.name <i>CustomTriggers</i> variableAttribute.value [<i>Custom1, Custom2, Custom3</i>]

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

Main (Test scenario)	
Charging Station	CSMS
Manual action: Use CSMS to trigger custom trigger Custom2 for <Configured evseld>	
2. The Test System responds with a TriggerMessageResponse with status <i>Accepted</i>	1. The CSMS sends a TriggerMessageRequest

Tool validations
* Step 1: Message: TriggerMessageRequest - requestedMessage <i>CustomTrigger</i> - customTrigger <i>Custom2</i> - evse.id <Configured evseld>
Post scenario validations: N/a

G Availability**TC_G_03_CSMS: Change Availability EVSE - Operative to inoperative**

Test case name	Change Availability EVSE - Operative to inoperative
Test case Id	TC_G_03_CSMS
Use case Id(s)	G03
Requirement(s)	N/a
System under test	CSMS
Description	This test case covers how the CSMS requests the Charging Station to change the availability of one of the EVSEs from Operative to Inoperative. An EVSE is considered Operative in any status other than Faulted and Unavailable.
Purpose	To verify if the CSMS is able to perform the change availability mechanism as described at the OCPP specification.
Prerequisite(s)	n/a

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

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>Unavailable</i> for <Configured evseld>	

Tool validations
N/a
Post scenario validations: - N/a

TC_G_04_CSMS: Change Availability EVSE - Inoperative to operative

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: <i>Unavailable</i> for <Configured evseld>
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 Test System responds with a ChangeAvailabilityResponse with status <i>Accepted</i>	1. The CSMS sends a ChangeAvailabilityRequest
3. The Test System notifies the CSMS about the current state of all connectors belonging to the specified EVSE (and optionally also from the EVSE itself). Message: StatusNotificationRequest - connectorStatus <i>Available</i> - evseld <i><Configured evseld></i> Message: NotifyEventRequest - trigger <i>Delta</i> - actualValue <i>"Available"</i> - component.name <i>"EVSE" / Connector</i> - component.evse.id <i><Configured evseld></i> - variable.name <i>"AvailabilityState"</i>	4. The CSMS responds accordingly.

Tool validations
* Step 1: Message ChangeAvailabilityRequest - operationalStatus <i>Operative</i> - evse.id <Configured evseld> - connectorId <i>omit</i>
Post scenario validations: - N/a

TC_G_05_CSMS: Change Availability Charging Station - Operative to inoperative

Test case name	Change Availability Charging Station - Operative to inoperative
Test case Id	TC_G_05_CSMS
Use case Id(s)	G04
Requirement(s)	N/a
System under test	CSMS
Description	<p>This test case describes how the CSMS requests the Charging Station to change the availability from operative to inoperative.</p> <p>A Charging Station is considered Operative when it is charging or ready for charging.</p> <p>A Charging Station is considered Inoperative when it does not allow any charging.</p>
Purpose	To verify if the CSMS is able to perform the change availability mechanism as described at the OCPP specification.
Prerequisite(s)	n/a

Before (Preparations)
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 Test System responds with a ChangeAvailabilityResponse with status <i>Accepted</i>	1. The CSMS sends a ChangeAvailabilityRequest
3. The Test System notifies the CSMS about the current state of all connectors Message: StatusNotificationRequest - connectorStatus <i>Unavailable</i> Message: NotifyEventRequest - trigger <i>Delta</i> - actualValue <i>"Unavailable"</i> - component.name <i>"Connector"</i> - variable.name <i>"AvailabilityState"</i>	4. The CSMS responds accordingly.

Tool validations
* Step 1: Message ChangeAvailabilityRequest - operationalStatus <i>Inoperative</i> - evseld <i>omit</i> - connectorId <i>omit</i>
Post scenario validations: - N/a

TC_G_06_CSMS: Change Availability Charging Station - Inoperative to operative

Test case name	Change Availability Charging Station - Inoperative to operative
Test case Id	TC_G_06_CSMS
Use case Id(s)	G04
Requirement(s)	N/a
System under test	CSMS
Description	<p>This test case describes how the CSMS requests the Charging Station to change the availability from inoperative to operative.</p> <p>A Charging Station is considered Operative when it is charging or ready for charging.</p> <p>A Charging Station is considered Inoperative when it does not allow any charging.</p>
Purpose	To verify if the CSMS is able to perform the change availability mechanism as described at the OCPP specification.
Prerequisite(s)	n/a

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): Charging Station set to <i>Unavailable</i> (Original status was Available)

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 Test System responds with a ChangeAvailabilityResponse with status <i>Accepted</i>	1. The CSMS sends a ChangeAvailabilityRequest
3. The Test System notifies the CSMS about the current state of all connectors. Message: StatusNotificationRequest - connectorStatus <i>Available</i> Message: NotifyEventRequest - trigger <i>Delta</i> - actualValue <i>"Available"</i> - component.name <i>"Connector"</i> - variable.name <i>"AvailabilityState"</i>	4. The CSMS responds accordingly.

Tool validations
* Step 1: Message ChangeAvailabilityRequest - operationalStatus <i>Operative</i> - evseld <i>omit</i> - connectorId <i>omit</i>
Post scenario validations: - N/a

TC_G_07_CSMS: Change Availability Connector - Operative to inoperative

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
<u>Manual Action</u> : Request the CSMS to change the availability of a Connector to Inoperative.	
2. The Test System responds with a ChangeAvailabilityResponse with status Accepted	1. The CSMS sends a ChangeAvailabilityRequest
3. The Test System notifies the CSMS about the current state of the connector. Message: StatusNotificationRequest - connectorStatus <i>Unavailable</i> - evseld <i><Configured evseld></i> - connectorId <i><Configured connectorId></i> Message: NotifyEventRequest - trigger <i>Delta</i> - actualValue <i>"Unavailable"</i> - component.name <i>"Connector"</i> - component.evse.id <i><Configured evseld></i> - component.evse.connectorId <i><Configured connectorId></i> - variable.name <i>"AvailabilityState"</i>	4. The CSMS responds accordingly.

Tool validations
* Step 1: Message ChangeAvailabilityRequest - operationalStatus <i>Inoperative</i> - evse.id <i><Configured evseld></i> - evse.connectorId <i><Configured connectorId></i>
Post scenario validations: N/a

TC_G_08_CSMS: Change Availability Connector - Inoperative to operative

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: <i>Unavailable</i> for <Configured connectorId>
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action</u> : Request the CSMS to change the availability of a Connector to Operative.	
2. The Test System responds with a ChangeAvailabilityResponse with status Accepted	1. The CSMS sends a ChangeAvailabilityRequest
3. The Test System notifies the CSMS about the current state of the connector. Message: StatusNotificationRequest - connectorStatus Available - evseld <Configured evseld> - connectorId <Configured connectorId> Message: NotifyEventRequest - trigger Delta - actualValue "Available" - component.name "Connector" - component.evse.id <Configured evseld> - component.evse.connectorId <Configured connectorId> - variable.name "AvailabilityState"	4. The CSMS responds accordingly.

Tool validations
* Step 1: Message ChangeAvailabilityRequest - operationalStatus <i>Operative</i> - evse.id <Configured evseld> - evse.connectorId <Configured connectorId>
Post scenario validations: N/a

TC_G_11_CSMS: Change Availability EVSE - With ongoing transaction

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 <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Note(s)</u> : Request the CSMS to change the availablitiy to inoperative	
2. The Test System responds with a ChangeAvailabilityResponse with status <i>Scheduled</i>	1. The CSMS sends a ChangeAvailabilityRequest
<u>Note(s)</u> : Wait for <Configured Transaction Duration>	
3. Execute Reusable State <i>StopAuthorized</i>	
4. Execute Reusable State <i>EVConnectedPostSession</i>	
5. Execute Reusable State <i>EVDisconnected</i>	
6. The Test System notifies the CSMS about the current state of all connectors with Message: StatusNotificationRequest - connectorStatus <i>Unavailable</i> - evseld <Configured evseld> OR Message: NotifyEventRequest - trigger <i>Delta</i> - actualValue "Unavailable" - component.name "Connector" - component.evse.id <Configured evseld> - variable.name "AvailabilityState"	7. The CSMS responds accordingly.

Tool validations
* Step 1: Message ChangeAvailabilityRequest - operationalStatus <i>Inoperative</i> - evse.id <Configured evseld> - connectorId omit
Post scenario validations: - A respond to report the state of a connector has been received for all connectors.

TC_G_14_CSMS: Change Availability Charging Station - With ongoing transaction

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 <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Note(s)</u> : Request the CSMS to change the availability of the station to inoperative	
2. The Test System responds with a ChangeAvailabilityResponse with status <i>Scheduled</i>	1. The CSMS sends a ChangeAvailabilityRequest
3. The Test System notifies the CSMS about the current state of all unoccupied connectors with Message: StatusNotificationRequest - connectorStatus <i>Unavailable</i>	4. The CSMS responds accordingly.
<u>Note(s)</u> : Wait for <Configured Transaction Duration>	
5. Execute Reusable State <i>StopAuthorized</i>	
6. Execute Reusable State <i>EVConnectedPostSession</i>	
7. Execute Reusable State <i>EVDisconnected</i>	
8. The Test System notifies the CSMS about the current state of the configured connector with Message: StatusNotificationRequest - connectorStatus <i>Unavailable</i>	9. The CSMS responds accordingly.

Tool validations
* Step 1: Message ChangeAvailabilityRequest - operationalStatus <i>Inoperative</i> - evseld <i>omit</i> - connectorId <i>omit</i>
Post scenario validations: - A respond to report the state of a connector has been received for all connectors.

TC_G_17_CSMS: Change Availability Connector - With ongoing transaction

Test case name	Change Availability Connector - With ongoing transaction
Test case Id	TC_G_17_CSMS
Use case Id(s)	G03
Requirement(s)	N/a
System under test	CSMS
Description	This test case covers how the CSMS requests the Charging Station to change the availability of one of the EVSEs from Operative to Inoperative. An EVSE is considered Operative in any status other than Faulted and Unavailable.
Purpose	To verify if the CSMS is able to send a change availability request during a transaction according to the mechanism as described at the OCPP specification.
Prerequisite(s)	n/a

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State: State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Note(s)</u> : Request the CSMS to change the availablitiy of one connector to inoperative	
2. The Test System responds with a ChangeAvailabilityResponse with status <i>Scheduled</i>	1. The CSMS sends a ChangeAvailabilityRequest
<u>Note(s)</u> : Wait for <Configured Transaction Duration>	
3. Execute Reusable State <i>StopAuthorized</i>	
4. Execute Reusable State <i>EVConnectedPostSession</i>	
5. Execute Reusable State <i>EVDisconnected</i>	
6. The Test System notifies the CSMS about the current state of all connectors with Message: StatusNotificationRequest - connectorStatus <i>Unavailable</i> - evseld <Configured evseld> - connectorId <Configured connectorId>	7. The CSMS responds accordingly.

Tool validations
* Step 1: Message ChangeAvailabilityRequest - operationalStatus <i>Inoperative</i> - 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.

TC_G_20_CSMS: Connector status Notification - Lock Failure

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
<p>1. The Test System sends a NotifyEventRequest with</p> <ul style="list-style-type: none">- eventData.trigger <i>Delta</i>- eventData.component.name <i>"ConnectorPlugRetentionLock"</i>- eventData.variable.name <i>"Problem"</i>- eventData.actualValue <i>"true"</i>	<p>2. The CSMS responds with a NotifyEventResponse</p>

Tool validations
N/a
Post scenario validations: - N/a

H Reservation

TC_H_01_CSMS: Reserve a specific EVSE - Accepted - Valid idToken

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 evseld.
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 <i>Reserved</i> for <Configured evseld>	

Tool validations
N/a
Post scenario validations: N/a

TC_H_07_CSMS: Reserve a specific EVSE - Reservation Ended / not used

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 evseld.
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
<u>Manual Action:</u> Trigger the CSMS to send a ReserveNowRequest for a specific EVSE.	
2. The Test System responds with a ReserveNowResponse With status <i>Accepted</i>	1. The CSMS sends a ReserveNowRequest with expiryDateTime <i>current time + <configured transaction duration></i>
3. The Test System notifies the CSMS about the current state of the connector(s) of the configured EVSE Message: StatusNotificationRequest with connectorStatus <i>Reserved</i> Message: NotifyEventRequest with trigger <i>Delta</i> actualValue <i>"Reserved"</i> component.name <i>"Connector"</i> variable.name <i>"AvailabilityState"</i>	4. The CSMS responds accordingly.
5. The Test System notifies the CSMS about the current state of the connector(s) of the configured EVSE Message: StatusNotificationRequest with connectorStatus <i>Available</i> Message: NotifyEventRequest with trigger <i>Delta</i> actualValue <i>"Available"</i> component.name <i>"Connector"</i> variable.name <i>"AvailabilityState"</i> <u>Note(s):</u> - The Test System waits until the provided expiryDateTime from step 1 expires before executing this step.	6. The CSMS responds accordingly.
7. The Test System sends a ReservationStatusUpdateRequest With reservationUpdateStatus <i>Expired</i> reservationId <i><id received at step 1></i>	8. The CSMS responds with a ReservationStatusUpdateResponse

Tool validations
<div>* Step 1: Message: ReserveNowRequest - evseld must be <i><Configured evseld></i> - connectorType must be omitted - idToken.idToken <i><Configured valid_idtoken_idtoken></i> - idToken.type <i><Configured valid_idtoken_type></i></div>
<div>Post scenario validations: N/a</div>

TC_H_08_CSMS: Reserve an unspecified EVSE - Accepted

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 evseld.
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
<u>Manual Action:</u> Trigger the CSMS to send a ReserveNowRequest for an unspecified EVSE.	
2. The Test System responds with a ReserveNowResponse with status Accepted	1. The CSMS sends a ReserveNowRequest
3. The Test System 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" <u>Note(s):</u> - The Test System will execute this step, if it is configured with only one EVSE.	4. The CSMS responds accordingly.

Tool validations
* Step 1: Message: ReserveNowRequest - evseld must be omitted - connectorType must be omitted - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type>
Post scenario validations: N/a

TC_H_14_CSMS: Reserve an unspecified EVSE - Amount of EVSEs available equals the amount of reservations

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 evseld.
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 Test System responds with a ReserveNowResponse with status <i>Accepted</i>	1. The CSMS sends a ReserveNowRequest <u>Note(s):</u> - This step needs to executed time the amount of EVSE configured for the Test System.
3. The Test System notifies the CSMS about the current state of all connectors Message: StatusNotificationRequest with connectorStatus <i>Reserved</i> Message: NotifyEventRequest with trigger <i>Delta</i> actualValue <i>"Reserved"</i> component.name <i>"Connector"</i> variable.name <i>"AvailabilityState"</i> <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 - evseld must be omitted - connectorType must be omitted - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type>

Tool validations
Post scenario validations: N/a

TC_H_15_CSMS: Reserve a connector with a specific type - Success

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 Test System responds with a ReserveNowResponse With status <i>Accepted</i>	1. The CSMS sends a ReserveNowRequest
3. The Test System notifies the CSMS about the current state of the connector Message: StatusNotificationRequest with connectorStatus <i>Reserved</i> Message: NotifyEventRequest with trigger <i>Delta</i> actualValue <i>"Reserved"</i> component.name <i>"Connector"</i> variable.name <i>"AvailabilityState"</i>	4. The CSMS responds accordingly.

Tool validations
* Step 1: Message: ReserveNowRequest - evseld must be omitted - connectorType must be <i><Configured connectorType></i> - idToken.idToken <i><Configured valid_idtoken_idtoken></i> - idToken.type <i><Configured valid_idtoken_type></i>
Post scenario validations: N/a

TC_H_17_CSMS: Cancel reservation of an EVSE - Success

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
<u>Manual Action:</u> Trigger the CSMS to send a <i>ReserveNowRequest</i> for a specific EVSE.	
2. The Test System responds with a ReserveNowResponse with status <i>Accepted</i>	1. The CSMS sends a ReserveNowRequest
3. The Test System notifies the CSMS about the current state of the connector(s) of the configured EVSE Message: StatusNotificationRequest with connectorStatus <i>Reserved</i> Message: NotifyEventRequest with trigger <i>Delta</i> actualValue <i>"Reserved"</i> component.name <i>"Connector"</i> variable.name <i>"AvailabilityState"</i>	4. The CSMS responds accordingly.
<u>Manual Action:</u> Trigger the CSMS to send a <i>CancelReservationRequest</i> for the reservation created at step 1.	
6. The Test System responds with a CancelReservationResponse With status <i>Accepted</i>	5. The CSMS sends a CancelReservationRequest
7. The Test System notifies the CSMS about the current state of the connector(s) of the configured EVSE Message: StatusNotificationRequest with connectorStatus <i>Available</i> Message: NotifyEventRequest with trigger <i>Delta</i> actualValue <i>"Available"</i> component.name <i>"Connector"</i> variable.name <i>"AvailabilityState"</i>	8. The CSMS responds accordingly.

Tool validations
<div>* Step 1: Message: ReserveNowRequest - evseld must be <Configured evseld> - 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</div>
<div>Post scenario validations: N/a</div>

TC_H_19_CSMS: Reserve a specific EVSE - Use a reserved EVSE with GroupId

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 <i>ReserveNowRequest</i> with a <i>groupIdToken</i> for a specific EVSE.	
2. The Test System responds with a ReserveNowResponse With status <i>Accepted</i>	1. The CSMS sends a ReserveNowRequest
3. The Test System notifies the CSMS about the current state of the connector(s) of the configured EVSE Message: StatusNotificationRequest with connectorStatus <i>Reserved</i> Message: NotifyEventRequest with trigger <i>Delta</i> actualValue <i>"Reserved"</i> component.name <i>"Connector"</i> variable.name <i>"AvailabilityState"</i>	4. The CSMS responds accordingly.

Tool validations
* Step 1: Message: ReserveNowRequest - evseld must be <i><Configured evseld></i> - connectorType must be omitted - groupIdToken must be provided - idToken.idToken <i><Configured valid_idtoken_idtoken></i> - idToken.type <i><Configured valid_idtoken_type></i>
Post scenario validations: N/a

TC_H_20_CSMS: Charging Station cancels reservation when Faulted

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 Test System.
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 Test System responds with a ReserveNowResponse With status <i>Accepted</i>	1. The CSMS sends a ReserveNowRequest
3. The Test System notifies the CSMS about the current state of the connector(s) of the configured EVSE Message: StatusNotificationRequest with connectorStatus <i>Reserved</i> Message: NotifyEventRequest with trigger <i>Delta</i> actualValue "Reserved" component.name "Connector" variable.name "AvailabilityState"	4. The CSMS responds accordingly.
5. The Test System notifies the CSMS about the current state of the connector(s) of the configured EVSE Message: StatusNotificationRequest with connectorStatus <i>Faulted</i> Message: NotifyEventRequest with trigger <i>Delta</i> actualValue "Faulted" component.name "Connector" variable.name "AvailabilityState"	6. The CSMS responds accordingly.
7. The Test System sends a ReservationStatusUpdateRequest With reservationUpdateStatus <i>Removed</i> reservationId <id received at step 1>	8. The CSMS responds with a ReservationStatusUpdateResponse

Tool validations
<div>* Step 1:</div> <div>Message: ReserveNowRequest</div> <div><div>- evseld must be <i><Configured evseld></i></div><div>- connectorType must be omitted</div><div>- idToken.idToken <i><Configured valid_idtoken_idtoken></i></div><div>- idToken.type <i><Configured valid_idtoken_type></i></div></div>
<div>Post scenario validations:</div> <div>N/a</div>

TC_H_22_CSMS: Reserve a specific EVSE - Configured to Reject

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 evseld.
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 Test System responds with a ReserveNowResponse with - status <i>Rejected</i>	1. The CSMS sends a ReserveNowRequest

Tool validations
N/a
Post scenario validations: N/a

I Tariff and Cost

TC_I_01_CSMS: Show EV Driver running total cost during charging - costUpdatedRequest

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 Test System 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 Test System 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 <i>EVConnectedPreSession</i>	
6. Execute Reusable State <i>EnergyTransferStarted</i>	
7. The Test System sends a TransactionEventRequest With triggerReason is <i>MeterValuePeriodic</i> eventType is <i>Updated</i> timestamp <The intervals between the timestamps of the received Meter Value messages equals the configured sampled Meter Values interval>. sampledValue.context is <i>Sample.Periodic</i> <u>Note(s):</u> - This step will be executed every <Configured sampled Meter Values interval> - The Test System will end the testcase after two MeterValues.	8. The CSMS responds with a TransactionEventResponse
10. The Test System responds with a CostUpdatedResponse	9. The CSMS sends a CostUpdatedRequest <u>Note(s):</u> - This step will be executed after every <i>TransactionEventResponse</i> , if the message did not contain a <i>totalCost</i> .

Tool validations
<p>* Step 2:</p> <p>Message AuthorizeResponse</p> <ul style="list-style-type: none">- idTokenInfo.status <i>Accepted</i> <p>* Step 4:</p> <p>Message TransactionEventResponse</p> <ul style="list-style-type: none">- idTokenInfo.status <i>Accepted</i>- totalCost <i><Optional></i> <p>* Step 7:</p> <p>Message (Optional) CostUpdatedRequest</p> <ul style="list-style-type: none">- transactionId <i><Generated TransactionId></i>
<p>Post scenario validations:</p> <ul style="list-style-type: none">- N/a

TC_I_02_CSMS: Show EV Driver Final Total Cost After Charging

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: - CSMS is configured with a tariff which is based on energy consumed.
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>EVConnectedPreSession</i>	
- The TransactionEventRequest contains the MeterValue field. - sampledValue[0].value 1000 - sampledValue[0].context <i>Transaction.Begin</i>	
2. Execute Reusable State <i>Authorized</i>	
3. Execute Reusable State <i>EnergyTransferStarted</i>	
4. Execute Reusable State <i>StopAuthorized</i>	
5. Execute Reusable State <i>EVConnectedPostSession</i>	
6. The Test System notifies the CSMS about the current state of the configured connector.	7. The CSMS responds accordingly.
Message: StatusNotificationRequest - connectorStatus <i>Available</i> Message: NotifyEventRequest - trigger <i>Delta</i> - actualValue <i>"Available"</i> - component.name <i>"Connector"</i> - variable.name <i>"AvailabilityState"</i>	
8. The Test System sends a TransactionEventRequest with	9. The CSMS responds with a TransactionEventResponse
- triggerReason <i>EVCommunicationLost</i> - eventType <i>Ended</i> - transactionInfo.chargingState <i>Idle</i> - transactionInfo.stoppedReason <i>EVDisconnected</i> - meterValue[0].sampledValue[0].value 6000 - meterValue[0].sampledValue[0].context <i>Transaction.End</i>	

Tool validations
* Step 9: Message TransactionEventResponse - totalCost <Not omitted>
Post scenario validations: - N/a

TC_I_101_CSMS: Set Default Tariff - startTimeOfDay, endTimeOfDay

Test case name	Set Default Tariff - startTimeOfDay, endTimeOfDay
Test case Id	TC_I_101_CSMS
Use case Id(s)	I07, I12
Requirement(s)	...
System under test	CSMS
Description	To set the default tariff on the charging station, for example for ad hoc charging, or when there is no driver-specific tariff.
Purpose	To verify if the CSMS is able to set a tariff with conditions.
Prerequisite(s)	<ul style="list-style-type: none"> - The CSMS supports local cost calculation by CS - The CSMS supports conditions in Tariffs - CS has TariffCtrlr.Enabled[Tariff] = true - CS has TariffCtrlr.MaxElements[Tariff] > 0 - CS has TariffCtrlr.ConditionsSupported[Tariff] = true

Before (Preparations)

Configuration State:

N/a

Memory State:

N/a

Reusable State:

N/a

Main (Test scenario)

Charging Station	CSMS
<u>Manual Action</u> : Request the CSMS to set a default tariff on EVSE 0 with <ul style="list-style-type: none"> - validFrom - Price for energy with condition on startTimeOfDay and endTimeOfDay 	
2. The Test System responds with SetDefaultTariffResponse with status Accepted	1. The CSMS sends a SetDefaultTariffRequest

Tool validations

* Step 1:

Message **SetDefaultTariffRequest**- **evseld** 0- **tariff.validFrom** must be <not omitted>- **tariff.energy.prices[0].priceKwh** must be <not omitted>- **tariff.energy.prices[0].conditions.startTimeOfDay** must be in format as per RFC 3339: time-hour ":" time-minute- **tariff.energy.prices[0].conditions.endTimeOfDay** must be in format as per RFC 3339: time-hour ":" time-minute

Post scenario validations:

N/a

TC_I_102_CSMS: Set Default Tariff - TariffMaxElements

Test case name	Set Default Tariff - TariffMaxElements
Test case Id	TC_I_102_CSMS
Use case Id(s)	I07, I11
Requirement(s)	...
System under test	CSMS
Description	To set the default tariff on the charging station, for example for ad hoc charging, or when there is no driver-specific tariff.
Purpose	To verify if the CSMS is able to process a response indicating TooManyElements.
Prerequisite(s)	<ul style="list-style-type: none"> - The CSMS supports local cost calculation by CS - CS has TariffCtrlr.Enabled[Tariff] = true - CS has TariffCtrlr.MaxElements[Tariff] > 0 - CS has TariffCtrlr.ConditionsSupported[Tariff] = true

Before (Preparations)

Configuration State:

N/a

Memory State:

N/a

Reusable State:

N/a

Main (Test scenario)

Charging Station	CSMS
<u>Manual Action:</u> _Request the CSMS to set a default tariff on EVSE 0 with <ul style="list-style-type: none"> - validFrom - Price for energy in peak hours - Price for energy other hours 	
2. The Test System responds with SetDefaultTariffResponse with status <i>TooManyElements</i>	1. The CSMS sends a SetDefaultTariffRequest

Tool validations

* Step 1:

Message **SetDefaultTariffRequest**- **evseld** 0- **tariff.validFrom** must be *<not omitted>*- **tariff.energy.prices[0].priceKwh** must be *<not omitted>*- **tariff.energy.prices[0].conditions.startTimeOfDay** must be in *format as per RFC 3339: time-hour ":" time-minute*- **tariff.energy.prices[0].conditions.endTimeOfDay** must be in *format as per RFC 3339: time-hour ":" time-minute*- **tariff.energy.prices[1].priceKwh** must be *<not omitted>*- **tariff.energy.prices[1].conditions.startTimeOfDay** must be in *format as per RFC 3339: time-hour ":" time-minute*- **tariff.energy.prices[1].conditions.endTimeOfDay** must be in *format as per RFC 3339: time-hour ":" time-minute*

Post scenario validations:

N/a

TC_I_105_CSMS: Set Default Tariff - TariffConditionsSupported is false

Test case name	Set Default Tariff - TariffConditionsSupported is false
Test case Id	TC_I_105_CSMS
Use case Id(s)	I07
Requirement(s)	...
System under test	CSMS
Description	To set the default tariff on the charging station, for example for ad hoc charging, or when there is no driver-specific tariff.
Purpose	To verify if the CSMS is able to process a response indicating ConditionNotSupported.
Prerequisite(s)	<ul style="list-style-type: none"> - The CSMS supports local cost calculation by CS - CS has TariffCtrlr.Enabled[Tariff] = true - CS has TariffCtrlr.MaxElements[Tariff] > 0 - CS has TariffCtrlr.ConditionsSupported[Tariff] = true

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action:</u> _Request the CSMS to set a default tariff on EVSE 0 with <ul style="list-style-type: none"> - validFrom - Price for energy in peak hours - Price for energy other hours 	
2. The Test System responds with SetDefaultTariffResponse with status <i>ConditionsNotSupported</i>	1. The CSMS sends a SetDefaultTariffRequest

Tool validations
* Step 1: Message SetDefaultTariffRequest <ul style="list-style-type: none"> - evseld 0 - tariff.validFrom must be <i><not omitted></i> - tariff.energy.prices[0].priceKwh must be <i><not omitted></i> - tariff.energy.prices[0].conditions.startTimeOfDay must be in <i>format as per RFC 3339: time-hour ":" time-minute</i> - tariff.energy.prices[0].conditions.endTimeOfDay must be in <i>format as per RFC 3339: time-hour ":" time-minute</i> - tariff.energy.prices[1].priceKwh must be <i><not omitted></i> - tariff.energy.prices[1].conditions.startTimeOfDay must be in <i>format as per RFC 3339: time-hour ":" time-minute</i> - tariff.energy.prices[1].conditions.endTimeOfDay must be in <i>format as per RFC 3339: time-hour ":" time-minute</i>
Post scenario validations: N/a

TC_I_106_CSMS: Set Default Tariff - validations

Test case name	Set Default Tariff - validations
Test case Id	TC_I_106_CSMS
Use case Id(s)	I07, I09
Requirement(s)	...
System under test	CSMS
Description	To set the default tariff on the charging station, for example for ad hoc charging, or when there is no driver-specific tariff.
Purpose	To verify if the CSMS supports configuring setting local cost calculation.
Prerequisite(s)	<ul style="list-style-type: none"> - The CSMS supports local cost calculation by CS - CS has TariffCtrlr.Enabled[Tariff] = true - CS has TariffCtrlr.MaxElements[Tariff] > 0 - CS has TariffCtrlr.ConditionsSupported[Tariff] = true

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action:</u> Request the CSMS to set a default tariff on EVSE 0 with <ul style="list-style-type: none"> - validFrom - Price for energy with a taxRate - Price for chargingTime with a taxRate - Price for idleTime with a taxRate - Price for fixedFee with a taxRate - Price for minCost with a taxRate - Price for maxCost with a taxRate - Price for reservationTime with a taxRate 	
2. The Test System responds with SetDefaultTariffResponse with status Accepted	1. The CSMS sends a SetDefaultTariffRequest
<u>Manual Action:</u> Request the CSMS to set a default tariff on EVSE 1 with <ul style="list-style-type: none"> - validFrom - Price for energy with a taxRate - Price for chargingTime with a taxRate - Price for idleTime with a taxRate - Price for fixedFee with a taxRate - Price for minCost with a taxRate - Price for maxCost with a taxRate - Price for reservationFixed with a taxRate 	
4. The Test System responds with SetDefaultTariffResponse with status Accepted	3. The CSMS sends a SetDefaultTariffRequest with
<u>Manual Action:</u> Request the CSMS to get the tariffs for EVSE 0	

Main (Test scenario)	
<p>6. The Test System responds with GetTariffsResponse with <i>status Accepted</i></p> <p>tariffAssignments[0].tariffId <value of tariffId from step 1></p> <p>tariffAssignments[0].tariffKind <i>DefaultTariff</i></p> <p>tariffAssignments[0].evselds [2]</p> <p>tariffAssignments[0].validFrom must be <not omitted></p> <p>tariffAssignments[1].tariffId <value of tariffId from step 3></p> <p>tariffAssignments[1].tariffKind <i>DefaultTariff</i></p> <p>tariffAssignments[1].evselds [1]</p> <p>tariffAssignments[1].validFrom must be <not omitted></p>	<p>5. The CSMS sends a GetTariffsRequest</p>

Tool validations
<p>* Step 1:</p> <p>Message SetDefaultTariffRequest</p> <ul style="list-style-type: none"> - evseld 0 - tariff.tariffId must be <not omitted> - tariff.validFrom must be <not omitted> - tariff.energy must be <not omitted> - tariff.energy.taxRates must be <not omitted> - tariff.chargingTime must be <not omitted> - tariff.chargingTime.taxRates must be <not omitted> - tariff.idleTime must be <not omitted> - tariff.idleTime.taxRates must be <not omitted> - tariff.fixedFee must be <not omitted> - tariff.fixedFee.taxRates must be <not omitted> - tariff.minCost must be <not omitted> - tariff.minCost.exclTax or tariff.minCost.inclTax must be <not omitted> (at least one must exist) - tariff.minCost.taxRates must be <not omitted> - tariff.maxCost must be <not omitted> - tariff.maxCost.exclTax or tariff.maxCost.inclTax must be <not omitted> (at least one must exist) - tariff.maxCost.taxRates must be <not omitted> - tariff.reservationTime must be <not omitted> - tariff.reservationTime.taxRates must be <not omitted>

Tool validations

* Step 3:

Message **SetDefaultTariffRequest**

- **evseld** 1
- **tariff.tariffId** must be *<Different from tariffId step 1>*
- **tariff.validFrom** must be *<not omitted>*
- **tariff.energy** must be *<not omitted>*
- **tariff.energy.taxRates** must be *<not omitted>*

- **tariff.chargingTime** must be *<not omitted>*
- **tariff.chargingTime.taxRates** must be *<not omitted>*

- **tariff.idleTime** must be *<not omitted>*
- **tariff.idleTime.taxRates** must be *<not omitted>*

- **tariff.fixedFee** must be *<not omitted>*
- **tariff.fixedFee.taxRates** must be *<not omitted>*

- **tariff.minCost** must be *<not omitted>*
- **tariff.minCost.exclTax** must be *<not omitted>*
- **tariff.minCost.inclTax** must be *<not omitted>*
- **tariff.minCost.taxRates** must be *<not omitted>*

- **tariff.maxCost** must be *<not omitted>*
- **tariff.maxCost.exclTax** must be *<not omitted>*
- **tariff.maxCost.inclTax** must be *<not omitted>*
- **tariff.maxCost.taxRates** must be *<not omitted>*

- **tariff.reservationFixed** must be *<not omitted>*
- **tariff.reservationFixed.taxRates** must be *<not omitted>*

* Step 5:

Message **GetTariffsRequest**

- **evseld** must be 0

Post scenario validations:

N/a

TC_I_109_CSMS: Receive Driver Tariff - Goodflow

Test case name	Receive Driver Tariff - Goodflow
Test case Id	TC_I_109_CSMS
Use case Id(s)	I08
Requirement(s)	I08.FR.01
System under test	CSMS
Description	To support receiving the driver-specific tariff to enable local cost calculation based on a tariff for this driver.
Purpose	To verify if the CSMS supports driver tariffs.
Prerequisite(s)	<ul style="list-style-type: none"> - The CSMS supports local cost calculation by CS - CS has TariffCtrlr.Enabled[Tariff] = true - CS has TariffCtrlr.MaxElements[Tariff] > 0

Before (Preparations)
Configuration State: CSMS is configured to provide a driver specific tariff for <Configured valid_idtoken_idtoken> that specifies: <ul style="list-style-type: none"> - currency: <not omitted> - energy price: 0.25 per kWh (ex VAT), vat 20% - idle price: 0.10 per minute (ex VAT), vat 20% - fixed fee: 0.50 (ex VAT), vat 20%
Memory State: N/a
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
1. The Test System sends an AuthorizeRequest with idToken.idToken <Configured valid_idtoken_idtoken>	2. The CSMS responds with AuthorizeResponse

Tool validations
<p>* Step 2:</p> <p>Message AuthorizeResponse</p> <ul style="list-style-type: none"> - idTokenInfo.status must be <i>Accepted</i> - tariff.tariffId must be <not omitted> - tariff.currency must be <not omitted> <ul style="list-style-type: none"> - tariff.energy.prices[0].priceKwh must be 0.25 - tariff.energy.taxRates[0].tax must be 20 - tariff.energy.taxRates[0].type must be VAT <ul style="list-style-type: none"> - tariff.idleTime.prices[0].priceMinute must be 0.10 - tariff.idleTime.taxRates[0].tax must be 20 - tariff.idleTime.taxRates[0].type must be VAT <ul style="list-style-type: none"> - tariff.fixedFee.prices[0].priceFixed must be 0.50 - tariff.fixedFee.taxRates[0].tax must be 20 - tariff.fixedFee.taxRates[0].type must be VAT
Post scenario validations: N/a

TC_I_110_CSMS: Clear Tariffs - DefaultTariff

Test case name	Clear Tariffs - DefaultTariff
Test case Id	TC_I_110_CSMS
Use case Id(s)	I10
Requirement(s)	...
System under test	Charging Station
Description	To clear previously set tariffs on the charging station.
Purpose	To verify if the CSMS is able to clear tariffs.
Prerequisite(s)	<ul style="list-style-type: none"> - The CSMS supports local cost calculation by CS - CS has TariffCtrlr.Enabled[Tariff] = true - CS has TariffCtrlr.MaxElements[Tariff] > 0

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action:</u> Request the CSMS to set a default tariff on EVSE 0.	
2. The Test System responds with SetDefaultTariffResponse with status Accepted	1. The CSMS sends a SetDefaultTariffRequest
<u>Manual Action:</u> Request the CSMS to clear only the tariff set in step 1.	
4. The Test System responds with ClearTariffsResponse with clearTariffsResult[0].tariffId <SetDefaultTariffRequest.tariff.tariffId of step 1> clearTariffsResult[0].status Accepted	3. The CSMS sends a ClearTariffsRequest
<u>Manual Action:</u> Request the CSMS to clear all tariffs.	
6. The Test System responds with ClearTariffsResponse with clearTariffsResult[0].tariffId <omitted> clearTariffsResult[0].status NoTariff	5. The CSMS sends a ClearTariffsRequest

Tool validations
<p>* Step 1: Message SetDefaultTariffRequest - evseld 0 - tariff.tariffId must be <not omitted></p> <p>* Step 3: Message ClearTariffsRequest - tariffIds[0] must be _<SetDefaultTariffRequest.tariff.tariffId of step 1></p> <p>* Step 5: Message ClearTariffsRequest - tariffIds must be <omitted></p>
Post scenario validations: N/a

TC_I_113_CSMS: Local Cost Calculation - Change transaction tariff - TariffMaxElements

Test case name	Local Cost Calculation - Change transaction tariff - TariffMaxElements
Test case Id	TC_I_113_CSMS
Use case Id(s)	I11
Requirement(s)	...
System under test	CSMS
Description	CSMS changes the tariff that is associated with a transaction. This may be needed when dealing with unexpected price changes.
Purpose	To verify if the CSMS is able to process a response indicating TooManyElements.
Prerequisite(s)	<ul style="list-style-type: none"> - The CSMS supports local cost calculation by CS. - CS has TariffCtrlr.Enabled[Tariff] = true - CS has TariffCtrlr.MaxElements[Tariff] > 0

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>EnergyTransferStarted</i>	
<u>Manual Action</u> : Request the CSMS to change the tariff of the transaction	
3. The Test System responds with ChangeTransactionTariffResponse with status <i>TooManyElements</i>	2. The CSMS sends a ChangeTransactionTariffRequest with

Tool validations
* Step 2: Message ChangeTransactionTariffRequest - transactionId must be <transaction id>
Post scenario validations: N/a

TC_I_114_CSMS: Local Cost Calculation - Change transaction tariff - ConditionNotSupported

Test case name	Local Cost Calculation - Change transaction tariff - TariffConditionsSupported is false
Test case Id	TC_I_114_CSMS
Use case Id(s)	I11
Requirement(s)	...
System under test	CSMS
Description	CSMS changes the tariff that is associated with a transaction. This may be needed when dealing with unexpected price changes.
Purpose	To verify if the CSMS is able to process a response indicating ConditionNotSupported.
Prerequisite(s)	<ul style="list-style-type: none"> - The CSMS supports local cost calculation by CS. - CS has TariffCtrlr.Enabled[Tariff] = true - CS has TariffCtrlr.MaxElements[Tariff] > 0 - CS has TariffCtrlr.ConditionsSupported[Tariff] = true

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>EnergyTransferStarted</i>	
<u>Manual Action:</u> Request the CSMS to change the tariff of the transaction with - <i>validFrom</i> - Price for energy with condition on <i>startTimeOfDay</i> and <i>endTimeOfDay</i>	
3. The Test System responds with ChangeTransactionTariffResponse with status <i>ConditionNotSupported</i>	2. The CSMS sends a ChangeTransactionTariffRequest with

Tool validations
* Step 2: Message ChangeTransactionTariffRequest <ul style="list-style-type: none"> - transactionId must be <transaction id> - tariff.validFrom must be <not omitted> - tariff.energy.prices[0].priceKwh must be <not omitted> - tariff.energy.prices[0].conditions.startTimeOfDay must be in format as per RFC 3339: time-hour ":" time-minute - tariff.energy.prices[0].conditions.endTimeOfDay must be in format as per RFC 3339: time-hour ":" time-minute
Post scenario validations: N/a

TC_I_115_CSMS: Local Cost Calculation - Change transaction tariff - Good flow

Test case name	Local Cost Calculation - Change transaction tariff - TariffConditionsSupported is true
Test case Id	TC_I_115_CSMS
Use case Id(s)	I11
Requirement(s)	...
System under test	CSMS
Description	CSMS changes the tariff that is associated with a transaction. This may be needed when dealing with unexpected price changes.
Purpose	To verify if the CSMS is able to change the tariff of a transaction with a tariff with conditions.
Prerequisite(s)	<ul style="list-style-type: none"> - The CSMS supports local cost calculation by CS. - CS has TariffCtrlr.Enabled[Tariff] = true - CS has TariffCtrlr.MaxElements[Tariff] > 0 - CS has TariffCtrlr.ConditionsSupported[Tariff] = true

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State EnergyTransferStarted	
<u>Manual Action:</u> Request the CSMS to change the tariff of the transaction with - validFrom - Price for energy with condition on startTimeOfDay and endTimeOfDay	
3. The Test System responds with ChangeTransactionTariffResponse with status Accepted	2. The CSMS sends a ChangeTransactionTariffRequest with

Tool validations
* Step 2: Message ChangeTransactionTariffRequest <ul style="list-style-type: none"> - transactionId must be <i><transaction id></i> - tariff.validFrom must be <i><not omitted></i> - tariff.energy.prices[0].priceKwh must be <i><not omitted></i> - tariff.energy.prices[0].conditions.startTimeOfDay must be in format as per RFC 3339: time-hour ":" time-minute - tariff.energy.prices[0].conditions.endTimeOfDay must be in format as per RFC 3339: time-hour ":" time-minute
Post scenario validations: N/a

TC_I_122_CSMS: Local Cost Calculation - Cost Details of Transaction

Test case name	Local Cost Calculation - Cost Details of Transaction
Test case Id	TC_I_122_CSMS
Use case Id(s)	I12
Requirement(s)	...
System under test	CSMS
Description	Charging Station calculates cost of the transaction locally and returns a break-down of the cost at end of the transaction for every charging period for which a different tariff element was active. CSMS can use this to generate invoices or Charge Detail Records for EMSPs.
Purpose	To verify if the CSMS is able to process the cost details sent by the Charging Station.
Prerequisite(s)	<ul style="list-style-type: none"> - The CSMS supports local cost calculation by CS. - CS has TariffCtrlr.Enabled[Tariff] = true - CS has TariffCtrlr.Enabled[Cost] = true - CS has TariffCtrlr.Enabled[RunningCost] = true - CS has TariffCtrlr.Interval[Cost] = 30 - CS has TariffCtrlr.MaxElements[Tariff] > 0 - CS has TariffCtrlr.ConditionsSupported[Tariff] = true

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 set a default tariff on EVSE 0 with <ul style="list-style-type: none">- Price for energy with a taxRate- Price for chargingTime with a taxRate- Price for idleTime with a taxRate- Price for fixedFee with a taxRate- Price for minCost with a taxRate- Price for maxCost with a taxRate	
2. The Test System responds with SetDefaultTariffResponse with status <i>Accepted</i>	1. The CSMS sends a SetDefaultTariffRequest
3. Execute Reusable State <i>EnergyTransferStarted</i>	

Main (Test scenario)	
<p>4. The Test System sends a TransactionEventRequest with eventType <i>Updated</i> triggerReason <i>RunningCost</i> costDetails.failureToCalculate <i><omitted></i> costDetails.failureReason <i><omitted></i> costDetails.totalCost.currency <i>EUR</i> costDetails.totalCost.typeOfCost <i>NormalCost</i> costDetails.totalCost.reservation <i><omitted></i> costDetails.totalCost.fixed.exclTax <i>15</i> costDetails.totalCost.fixed.inclTax <i>15.75</i> costDetails.totalCost.fixed.taxRates[0].type <i>t5_0</i> costDetails.totalCost.fixed.taxRates[0].tax <i>5.0</i> costDetails.chargingPeriods[0].tariffId <i><SetDefaultTariffRequest.tariff.tariffId of step 1></i> costDetails.chargingPeriods[0].startPeriod <i><txStartDateTime></i> costDetails.chargingPeriods[0].dimensions[0].type <i>ChargingTime</i> costDetails.chargingPeriods[0].dimensions[0].volume <i>15</i> costDetails.chargingPeriods[0].dimensions[1].type <i>Energy</i> costDetails.chargingPeriods[0].dimensions[1].volume <i>12</i></p>	<p>5. The CSMS responds with a TransactionEventResponse</p>

Tool validations
<p>* Step 1: Message SetDefaultTariffRequest - evseld <i>0</i></p> <p>- tariff.energy must be <i><not omitted></i> - tariff.energy.taxRates must be <i><not omitted></i></p> <p>- tariff.chargingTime must be <i><not omitted></i> - tariff.chargingTime.taxRates must be <i><not omitted></i></p> <p>- tariff.idleTime must be <i><not omitted></i> - tariff.idleTime.taxRates must be <i><not omitted></i></p> <p>- tariff.fixedFee must be <i><not omitted></i> - tariff.fixedFee.taxRates must be <i><not omitted></i></p> <p>- tariff.minCost must be <i><not omitted></i> - tariff.minCost.exclTax must be <i><not omitted></i> - tariff.minCost.inclTax must be <i><not omitted></i> - tariff.minCost.taxRates must be <i><not omitted></i></p> <p>- tariff.maxCost must be <i><not omitted></i> - tariff.maxCost.exclTax must be <i><not omitted></i> - tariff.maxCost.inclTax must be <i><not omitted></i> - tariff.maxCost.taxRates must be <i><not omitted></i></p> <p>Post scenario validations: N/a</p>

J Meter Values

TC_J_01_CSMS: Clock-aligned Meter Values - No transaction ongoing

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
<p>1. The Test System notifies the CSMS about its measured Meter Values.</p> <p>Message: MeterValuesRequest</p> <ul style="list-style-type: none"> - timestamp <The intervals between the timestamps of the received Meter Value messages equals the _<Configured clock-aligned Meter Values interval>. - sampledValue.context is <i>Sample.Clock</i> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none"> - timestamp <The intervals between the timestamps of the received Meter Value messages equals the _<Configured clock-aligned Meter Values interval>. - trigger is <i>Periodic</i> - component.name is <i>FiscalMetering</i> <p><u>Note(s):</u></p> <ul style="list-style-type: none"> - This step will be executed every _<Configured clock-aligned Meter Values interval> - This step will be executed for <i>evseld=0</i> and all configured EVSE. - The Test System will end the testcase after it has send three Meter Value messages. 	<p>2. The CSMS responds accordingly.</p>

Tool validations
N/a
Post scenario validations: N/a

TC_J_02_CSMS: Clock-aligned Meter Values - Transaction ongoing

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 <i>EnergyTransferStarted</i> for <Configured evseld>

Main (Test scenario)	
Charging Station	CSMS
<p>1. The Test System notifies the CSMS about its measured Meter Values.</p> <p>Message: MeterValuesRequest</p> <ul style="list-style-type: none"> - timestamp <The intervals between the timestamps of the received Meter Value messages equals the configured clock-aligned Meter Values interval>. - sampledValue.context is <i>Sample.Clock</i> <p>Message: NotifyEventRequest</p> <ul style="list-style-type: none"> - timestamp <The intervals between the timestamps of the received Meter Value messages equals the configured clock-aligned Meter Values interval>. - trigger is <i>Periodic</i> - component.name is <i>FiscalMetering</i> <p><u>Note(s):</u></p> <ul style="list-style-type: none"> - This step will be executed every _<Configured clock-aligned Meter Values interval> - This step will be executed for evseld=0 and all configured idle EVSE. 	<p>2. The CSMS responds accordingly.</p>

Main (Test scenario)	
<p>3. The Test System sends a TransactionEventRequest With triggerReason is <i>MeterValueClock</i> eventType is <i>Updated</i> timestamp <i><The intervals between the timestamps of the received Meter Value messages equals the configured clock-aligned Meter Values interval></i>. sampledValue.context is <i>Sample.Clock</i></p> <p><u>Note(s):</u> - <i>This step will be executed every _<Configured clock-aligned Meter Values interval></i> - <i>The Test System will end the testcase after the _<Configured transaction duration> is reached._</i></p>	<p>4. The CSMS responds with a TransactionEventResponse</p>
Tool validations	
N/a	
Post scenario validations:	
N/a	

TC_J_03_CSMS: Clock-aligned Meter Values - EventType Ended

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 <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
<p>1. Execute Reusable State <i>EVDIsconnected</i></p> <ul style="list-style-type: none">- The TransactionEventRequest containing eventType <i>Ended</i> contains the MeterValue field.- timestamp <i><The intervals between the timestamps of the received Meter Value messages equals the configured value at configured clock-aligned Tx ended Meter Values interval></i>.- sampledValue.context is <i>Sample.Clock</i> AND the last one has <i>Transaction.End</i> <p><u>Note(s):</u></p> <ul style="list-style-type: none">- <i>This step will be executed after the _<Configured transaction duration> is reached._</i>- <i>This causes the transaction to stop.</i>	

Tool validations
N/a
Post scenario validations: N/a

TC_J_04_CSMS: Clock-aligned Meter Values - Signed

Test case name	Clock-aligned Meter Values - Signed
Test case Id	TC_J_04_CSMS
Use case Id(s)	J01
Requirement(s)	J01.FR.21
System under test	CSMS
Description	The Charging Station samples the electrical meter or other sensor/transducer hardware to provide information about its Meter Values. Depending on configuration settings, the Charging Station will send Meter Values.
Purpose	To verify if the CSMS is able to handle a Charging Station sending clock-aligned Meter Values, when a transaction ends.
Prerequisite(s)	N/a

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
<p>1. Execute Reusable State <i>EVDIsconnected</i></p> <ul style="list-style-type: none">- The TransactionEventRequest containing eventType <i>Ended</i> contains the MeterValue field.- timestamp <i><The intervals between the timestamps of the received Meter Value messages equals the configured value at configured clock-aligned Tx ended Meter Values interval></i>.- sampledValue.context is <i>Sample.Clock</i> AND the last one has <i>Transaction.End</i>- sampledValue.signedMeterValue is <i><Generated SignedMeterValueType></i> <p><u>Note(s):</u></p> <ul style="list-style-type: none">- <i>This step will be executed after the _<Configured transaction duration> is reached._</i>- <i>This causes the transaction to stop.</i>	

Tool validations
N/a
Post scenario validations: N/a

TC_J_07_CSMS: Sampled Meter Values - EventType Started - EVSE known

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
<p>1. Execute Reusable State <i>EVConnectedPreSession</i></p> <p>- The TransactionEventRequest contains the MeterValue field.</p> <p>- sampledValue.context is <i>Transaction.Begin</i></p>	

Tool validations
N/a
Post scenario validations: N/a

TC_J_08_CSMS: Sampled Meter Values - Context Transaction.Begin - EVSE not known

Test case name	Sampled Meter Values - Context Transaction.Begin - EVSE not known
Test case Id	TC_J_08_CSMS
Use case Id(s)	J02
Requirement(s)	J02.FR.19
System under test	CSMS
Description	The Charging Station samples the electrical meter or other sensor/transducer hardware to provide information about its Meter Values. Depending on configuration settings, the Charging Station will send Meter Values.
Purpose	To verify if the CSMS is able to handle a Charging Station sending start sampled Meter Values, when a transaction starts.
Prerequisite(s)	N/a

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

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>Authorized</i>	
2. Execute Reusable State <i>EVConnectedPreSession</i>	
<div>- The TransactionEventRequest contains the MeterValue field.</div> <div>- sampledValue.context is <i>Transaction.Begin</i></div>	
3. Execute Reusable State <i>EnergyTransferStarted</i>	

Tool validations
N/a
Post scenario validations: N/a

TC_J_09_CSMS: Sampled Meter Values - EventType Updated

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 <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
<p>1. The Test System sends a TransactionEventRequest With triggerReason is <i>MeterValuePeriodic</i> eventType is <i>Updated</i> timestamp <The intervals between the timestamps of the received Meter Value messages equals the configured sampled Meter Values interval>. sampledValue.context is <i>Sample.Periodic</i></p> <p><u>Note(s):</u> - This step will be executed every _<Configured sampled Meter Values interval> - The Test System will end the testcase after three MeterValues.</p>	<p>2. The CSMS responds with a TransactionEventResponse</p>

Tool validations
N/a
Post scenario validations: N/a

TC_J_10_CSMS: Sampled Meter Values - EventType Ended

Test case name	Sampled Meter Values - EventType Ended
Test case Id	TC_J_10_CSMS
Use case Id(s)	J02
Requirement(s)	J02.FR.19
System under test	CSMS
Description	The Charging Station samples the electrical meter or other sensor/transducer hardware to provide information about its Meter Values. Depending on configuration settings, the Charging Station will send Meter Values.
Purpose	To verify if the CSMS is able to handle a Charging Station sending sampled Meter Values, when a transaction ends.
Prerequisite(s)	N/a

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
<p>1. Execute Reusable State <i>EVDIsconnected</i></p> <ul style="list-style-type: none">- The TransactionEventRequest containing eventType <i>Ended</i> contains the MeterValue field.- timestamp <i><The intervals between the timestamps of the received Meter Value messages equals the configured value at configured clock-aligned Tx ended Meter Values interval></i>.- sampledValue.context is <i>Sample.Periodic</i> AND the last one has <i>Transaction.End</i> <p><u>Note(s):</u></p> <ul style="list-style-type: none">- <i>This step will be executed after the _<Configured transaction duration> is reached._</i>- <i>This causes the transaction to stop.</i>	

Tool validations
N/a
Post scenario validations: N/a

TC_J_11_CSMS: Sampled Meter Values - Signed

Test case name	Sampled Meter Values - Signed
Test case Id	TC_J_11_CSMS
Use case Id(s)	J02
Requirement(s)	J02.FR.21
System under test	CSMS
Description	The Charging Station samples the electrical meter or other sensor/transducer hardware to provide information about its Meter Values. Depending on configuration settings, the Charging Station will send Meter Values.
Purpose	To verify if the CSMS is able to handle a Charging Station sending sampled Meter Values, when a transaction ends.
Prerequisite(s)	N/a

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
<p>1. Execute Reusable State <i>EVDIsconnected</i></p> <ul style="list-style-type: none">- The TransactionEventRequest containing eventType <i>Ended</i> contains the MeterValue field.- timestamp <i><The intervals between the timestamps of the received Meter Value messages equals the configured value at configured clock-aligned Tx ended Meter Values interval></i>.- sampledValue.context is <i>Sample.Periodic</i> AND the last one has <i>Transaction.End</i>- sampledValue.signedMeterValue is <i><Generated SignedMeterValueType></i> <p><u>Note(s):</u></p> <ul style="list-style-type: none">- <i>This step will be executed after the _<Configured transaction duration> is reached._</i>- <i>This causes the transaction to stop.</i>	

Tool validations
N/a
Post scenario validations: N/a

K Smart Charging

TC_K_01_CSMS: Set Charging Profile - TxDefaultProfile - Specific EVSE

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Test System responds with a SetChargingProfileResponse with status <i>Accepted</i>	1. The CSMS sends a SetChargingProfileRequest with - chargingProfile.id <Configured chargingProfileId>

Tool validations
<p>* Step 1:</p> <p>Message SetChargingProfileRequest</p> <p>evseld <Configured evseld> AND</p> <p>chargingProfile.stackLevel <Configured stackLevel> AND</p> <p>chargingProfile.chargingProfilePurpose TxDefaultProfile AND</p> <p>chargingProfile.chargingProfileKind Absolute AND</p> <p>chargingProfile.validFrom now AND</p> <p>chargingProfile.validTo now + <Configured Charging Schedule Duration> AND</p> <p>chargingProfile.chargingSchedule.startSchedule now AND</p> <p>chargingProfile.chargingSchedule.chargingRateUnit <Configured chargingRateUnit> AND</p> <p>chargingProfile.chargingSchedule.duration <Configured duration> AND</p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod.startPeriod 0 AND</p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod.limit 6.0 or 6000.0 AND</p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod.numberPhases <Configured numberPhases> where <Configured numberPhases> not 3 OR</p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod.numberPhases <Configured numberPhases> or <omit> where <Configured numberPhases> 3</p> <p>Post scenario validations:</p> <p>- N/a</p>

TC_K_02_CSMS: Set Charging Profile - TxProfile without ongoing transaction on the specified EVSE

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Test System responds with a SetChargingProfileResponse with status <i>Rejected</i>	1. The CSMS sends a SetChargingProfileRequest - chargingProfile.id < <i>Configured chargingProfileId</i> >

Tool validations
<p>* Step 1:</p> <p>Message SetChargingProfileRequest</p> <ul style="list-style-type: none"> - evseld <Configured evseld> AND - chargingProfile.chargingProfilePurpose TxProfile AND - chargingProfile.stackLevel <Configured stackLevel> AND - chargingProfile.chargingProfileKind Relative AND - chargingProfile.chargingSchedule.startSchedule must be omitted 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 <p>Post scenario validations:</p> <ul style="list-style-type: none"> - N/a

TC_K_03_CSMS: Set Charging Profile - ChargingStationMaxProfile

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

Reusable State:

N/a

Main (Test scenario)

Charging Station	CSMS
2. The Test System responds with a SetChargingProfileResponse with status Accepted	1. The CSMS sends a SetChargingProfileRequest - chargingProfile.id <Configured chargingProfileId>

Tool validations

* Step 1:

Message **SetChargingProfileRequest****evseld** 0 AND**chargingProfile.stackLevel** <Configured stackLevel> AND**chargingProfile.chargingProfilePurpose** *ChargingStationMaxProfile* AND**chargingProfile.chargingProfileKind** *Absolute***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.chargingSchedule.startSchedule** <Not omitted>

Post scenario validations:

- N/a

TC_K_04_CSMS: Replace charging profile - With chargingProfileId

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Test System 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 Test System 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
<p>* Step 1:</p> <p>Message SetChargingProfileRequest</p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod.startPeriod 0 AND</p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod.limit 8.0 (A) or 8000.0 (W)</p> <p>The chargingSchedule contains only one chargingSchedulePeriod</p> <p>* Step 3:</p> <p>Message SetChargingProfileRequest</p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod.startPeriod 0 AND</p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod.limit 6.0 (A) or 6000.0 (W)</p> <p>The chargingSchedule contains only one chargingSchedulePeriod</p> <p>* Step 1/3:</p> <p>Message SetChargingProfileRequest</p> <p>chargingProfile.id <Same id for both chargingProfiles></p> <p>chargingProfile.chargingSchedule.startSchedule must NOT be omitted.</p> <p>chargingProfile.stackLevel <Configured stackLevel> AND</p> <p>chargingProfile.chargingProfilePurpose <Equal value for both profiles> AND (TxDefaultProfile OR ChargingStationMaxProfile)</p> <p>chargingProfile.chargingProfileKind <Equal value for both profiles> AND</p> <p>If chargingProfile.chargingProfilePurpose is TxDefaultProfile then chargingProfile.chargingProfileKind must be Absolute OR Recurring</p> <p>If chargingProfile.chargingProfilePurpose is ChargingStationMaxProfile then chargingProfile.chargingProfileKind must be Absolute</p> <p>If chargingProfile.chargingProfileKind is Recurring then chargingProfile.recurrencyKind must NOT be omitted, else omitted</p> <p>The received Charging Profiles must comply with the requirements defined at part 2 specification.</p>
<p>Post scenario validations:</p> <p>- N/a</p>

TC_K_05_CSMS: Clear Charging Profile - With chargingProfileId

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 Test System responds with a GetChargingProfilesResponse with status <i>Accepted</i> Test System sends a ReportChargingProfilesRequest CSMS responds with a ReportChargingProfilesResponse
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Test System responds with a ClearChargingProfileResponse with status <i>Accepted</i>	1. The CSMS sends a ClearChargingProfileRequest with chargingProfileId <i><Generated chargingProfileId></i> AND chargingProfileCriteria <i>omit</i>

Tool validations
* Step 1: Message ClearChargingProfileRequest chargingProfileId <Generated chargingProfileId> AND chargingProfileCriteria omit
Post scenario validations: - N/a

TC_K_06_CSMS: Clear Charging Profile - With stackLevel/purpose combination for one profile

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Test System responds with a ClearChargingProfileResponse with status <i>Accepted</i>	1. The CSMS sends a ClearChargingProfileRequest with chargingProfilePurpose <i>TxDefaultProfile</i> AND evseld <i><Configured evseld></i> AND stackLevel <i><Configured stackLevel></i>

Tool validations
* Step 1: Message ClearChargingProfileRequest chargingProfileCriteria.chargingProfilePurpose <i>TxDefaultProfile</i> AND chargingProfileCriteria.stackLevel <i><Configured stackLevel></i> AND chargingProfileCriteria.evseld <i><Configured evseld></i> AND chargingProfileId must be omitted.
Post scenario validations: - N/a

TC_K_08_CSMS: Clear Charging Profile - Without previous charging profile

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Test System responds with a ClearChargingProfileResponse with status <i>Unknown</i>	1. The CSMS sends a ClearChargingProfileRequest with chargingProfilePurpose <i>TxDefaultProfile</i> AND evseld <i><Configured evseld></i> AND stackLevel <i><Configured stackLevel></i>

Tool validations
<p>* Step 1:</p> <p>Message ClearChargingProfileRequest</p> <p>chargingProfileCriteria.chargingProfilePurpose <i>TxDefaultProfile</i> AND</p> <p>chargingProfileCriteria.stackLevel <i><Configured stackLevel></i> AND</p> <p>chargingProfileCriteria.evseld <i><Configured evseld></i> AND</p> <p>chargingProfileId must be omitted.</p>
<p>Post scenario validations:</p> <p>- N/a</p>

TC_K_10_CSMS: Set Charging Profile - TxDefaultProfile - All EVSE

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Test System responds with a SetChargingProfileResponse with status <i>Accepted</i>	1. The CSMS sends a SetChargingProfileRequest with - chargingProfile.id < <i>Configured chargingProfileId</i> >

Tool validations
<p>* Step 1:</p> <p>Message SetChargingProfileRequest</p> <p>evseld 0 AND</p> <p>chargingProfile.stackLevel <Configured stackLevel> AND</p> <p>chargingProfile.chargingProfilePurpose TxDefaultProfile AND</p> <p>chargingProfile.chargingProfileKind Absolute AND</p> <p>chargingProfile.chargingSchedule.startSchedule <Not omitted> AND</p> <p>chargingProfile.chargingSchedule.chargingRateUnit <Configured ChargingRateUnit> AND</p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod.startPeriod 0 AND</p> <p>chargingProfile.chargingSchedule.duration <Configured duration></p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod.limit 6.0 or 6000.0 AND</p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod.numberPhases <Configured numberPhases> where <Configured numberPhases> not 3 OR</p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod.numberPhases <Configured numberPhases> or <omit> where <Configured numberPhases> 3</p>
Post scenario validations:
- N/a

TC_K_15_CSMS: Set Charging Profile - Not Supported

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

Reusable State:

N/a

Main (Test scenario)

Charging Station	CSMS
2. The Test System responds with RPC Framework: CALLERROR: NotSupported.	<p>1. The CSMS sends a SetChargingProfileRequest with:</p> <p>evseld <Configured evseld> AND</p> <p>chargingProfile.stackLevel <Configured stackLevel> AND</p> <p>chargingProfile.chargingProfilePurpose TxDefaultProfile AND</p> <p>chargingProfile.chargingProfileKind Absolute AND</p> <p>chargingProfile.chargingSchedule.startSchedule <Not omitted> AND</p> <p>chargingProfile.chargingSchedule.chargingRateUnit <Configured ChargingRateUnit> AND</p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod.startPeriod 0 AND</p> <p>chargingProfile.chargingSchedule.duration <Configured duration></p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod.limit 6.0 or 6000.0 AND</p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod.numberPhases <Configured numberPhases> where <Configured numberPhases> not 3 OR</p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod.numberPhases <Configured numberPhases> or <omit> where <Configured numberPhases> 3</p>

Tool validations
<p>* Step 1:</p> <p>Message SetChargingProfileRequest</p> <p>evseld <Configured evseld> AND</p> <p>chargingProfile.stackLevel <Configured stackLevel> AND</p> <p>chargingProfile.chargingProfilePurpose <i>TxDefaultProfile</i> AND</p> <p>chargingProfile.chargingProfileKind <i>Absolute</i> AND</p> <p>chargingProfile.chargingSchedule.startSchedule <Not omitted> AND</p> <p>chargingProfile.chargingSchedule.chargingRateUnit <Configured ChargingRateUnit> AND</p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod.startPeriod 0 AND</p> <p>chargingProfile.chargingSchedule.duration <Configured duration></p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod.limit 6.0 or 6000.0 AND</p> <p>chargingProfile.chargingSchedule.chargingSchedulePeriod.numberPhases <Configured numberPhases></p>
<p>Post scenario validations:</p> <p>- N/a</p>

TC_K_19_CSMS: Set Charging Profile - ChargingProfileKind is Recurring

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Test System responds with a SetChargingProfileResponse with - status <i>Accepted</i>	1. The CSMS sends a SetChargingProfileRequest

Tool validations
<p>* Step 1:</p> <p>Message SetChargingProfileRequest</p> <ul style="list-style-type: none"> - evseld <Configured evseld> AND - chargingProfile.stackLevel <Configured stackLevel> AND - chargingProfile.chargingProfilePurpose TxDefaultProfile AND - chargingProfile.chargingSchedule.chargingSchedulePeriod.startPeriod 0 AND - chargingProfile.chargingProfileKind Recurring AND - chargingProfile.recurrencyKind <Configured recurrencyKind>
<p>Post scenario validations:</p> <ul style="list-style-type: none"> - N/a

TC_K_29_CSMS: Get Charging Profile - Evseld 0

Test case name	Get Charging Profile - Evseld 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
Reusable State: <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
2. The Test System responds with a GetChargingProfilesResponse with - <i>status Accepted</i>	1. The CSMS sends a GetChargingProfilesRequest with - <i>evseld 0</i>
3. The Test System sends a ReportChargingProfilesRequest with - <i>requestId <Received requestId></i>	4. The CSMS responds with a ReportChargingProfilesResponse

Tool validations
<p>* Step 1:</p> <p>Message GetChargingProfilesRequest</p> <p>- <i>evseld 0</i> AND</p> <p>- <i>chargingProfile.chargingProfilePurpose <Configured chargingProfilePurpose></i> AND</p> <p><u>Note</u>: <i>chargingProfilePurpose</i> is included, because the <i>chargingProfile</i> field is required and may not be left empty.</p> <p>- <i>chargingProfile.stackLevel</i> must be omitted AND</p> <p>- <i>chargingProfile.chargingLimitSource</i> must be omitted AND</p> <p>- <i>chargingProfile.chargingProfileId</i> must be omitted</p>
Post scenario validations:
- N/a

TC_K_30_CSMS: Get Charging Profile - Evseld > 0

Test case name	Get Charging Profile - Evseld > 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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Test System responds with a GetChargingProfilesResponse with - status <i>Accepted</i>	1. The CSMS sends a GetChargingProfilesRequest
3. The Test System sends a ReportChargingProfilesRequest with - requestId <i><Received requestId></i>	4. The CSMS responds with a ReportChargingProfilesResponse

Tool validations
<p>* Step 1:</p> <p>Message GetChargingProfilesRequest</p> <ul style="list-style-type: none"> - evseld <i><Configured evseld></i> AND - chargingProfile.chargingProfilePurpose <i><Configured chargingProfilePurpose></i> AND <p><u>Note</u>: <i>chargingProfilePurpose</i> is included, because the <i>chargingProfile</i> field is required and may not be left empty.</p> <ul style="list-style-type: none"> - chargingProfile.stackLevel must be omitted AND - chargingProfile.chargingLimitSource must be omitted AND - chargingProfile.chargingProfileId must be omitted
Post scenario validations:
- N/a

TC_K_31_CSMS: Get Charging Profile - No Evseld

Test case name	Get Charging Profile - No Evseld
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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Test System responds with a GetChargingProfilesResponse with - status <i>Accepted</i>	1. The CSMS sends a GetChargingProfilesRequest with - requestId <i><Received requestId></i>
3. The Test System sends a ReportChargingProfilesRequest with - requestId <i><Received requestId></i> AND - tb <i>true</i> AND - evseld <i>i</i>	4. The CSMS responds with a ReportChargingProfilesResponse
Note(s): - Step 3 and 4 are repeated for every evse	

Tool validations
<p>* Step 1:</p> <p>Message GetChargingProfilesRequest</p> <ul style="list-style-type: none"> - evseld must be omitted AND - chargingProfile.chargingProfilePurpose <i><Configured chargingProfilePurpose></i> AND <p><u>Note:</u> <i>chargingProfilePurpose is included, because the chargingProfile field is required and may not be left empty.</i></p> <ul style="list-style-type: none"> - chargingProfile.stackLevel must be omitted AND - chargingProfile.chargingLimitSource must be omitted AND - chargingProfile.chargingProfileId must be omitted
<p>Post scenario validations:</p> <ul style="list-style-type: none"> - N/a

TC_K_32_CSMS: Get Charging Profile - chargingProfileId

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Test System responds with a GetChargingProfilesResponse with - status <i>Accepted</i>	1. The CSMS sends a GetChargingProfilesRequest - chargingProfileId <Received <i>chargingProfileId</i> >
3. The Test System sends a ReportChargingProfilesRequest with - requestId <i>Generated Id</i>	4. The CSMS responds with a ReportChargingProfilesResponse

Tool validations
<p>* Step 1:</p> <p>Message GetChargingProfilesRequest</p> <ul style="list-style-type: none"> - evseld must be omitted AND - chargingProfile.chargingProfilePurpose must be omitted AND - chargingProfile.stackLevel must be omitted AND - chargingProfile.chargingLimitSource must be omitted AND - chargingProfile.chargingProfileId <received <i>chargingProfileId</i>>
<p>Post scenario validations:</p> <ul style="list-style-type: none"> - N/a

TC_K_33_CSMS: Get Charging Profile - Evseld > 0 + stackLevel

Test case name	Get Charging Profile - Evseld > 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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Test System responds with a GetChargingProfilesResponse with - status <i>Accepted</i>	1. The CSMS sends a GetChargingProfilesRequest
3. The Test System sends a ReportChargingProfilesRequest with - requestId <i>Generated Id</i>	4. The CSMS responds with a ReportChargingProfilesResponse

Tool validations
<p>* Step 1:</p> <p>Message GetChargingProfilesRequest</p> <ul style="list-style-type: none"> - evseld <Configured evseld> AND - chargingProfile.stackLevel <Configured stackLevel> AND - chargingProfile.chargingProfilePurpose must be omitted AND - chargingProfile.chargingLimitSource must be omitted AND - chargingProfile.chargingProfileId must be omitted
Post scenario validations:
- N/a

TC_K_34_CSMS: Get Charging Profile - Evseld > 0 + chargingLimitSource

Test case name	Get Charging Profile - Evseld > 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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Test System responds with a GetChargingProfilesResponse with - status <i>Accepted</i>	1. The CSMS sends a GetChargingProfilesRequest
3. The Test System sends a ReportChargingProfilesRequest with - requestId <i>Generated Id</i>	4. The CSMS responds with a ReportChargingProfilesResponse

Tool validations
<p>* Step 1:</p> <p>Message GetChargingProfilesRequest</p> <ul style="list-style-type: none"> - evseld <Configured evseld> AND - chargingProfile.chargingLimitSource <Configured chargingLimitSource> AND - chargingProfile.stackLevel must be omitted AND - chargingProfile.chargingProfilePurpose must be omitted AND - chargingProfile.chargingProfileId must be omitted
Post scenario validations:
- N/a

TC_K_35_CSMS: Get Charging Profile - Evseld > 0 + chargingProfilePurpose

Test case name	Get Charging Profile - Evseld > 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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Test System responds with a GetChargingProfilesResponse with - status <i>Accepted</i>	1. The CSMS sends a GetChargingProfilesRequest
3. The Test System sends a ReportChargingProfilesRequest with - requestId <i>Generated Id</i>	4. The CSMS responds with a ReportChargingProfilesResponse

Tool validations
<p>* Step 1:</p> <p>Message GetChargingProfilesRequest</p> <ul style="list-style-type: none"> - evseld <Configured evseld> AND - chargingProfile.chargingProfilePurpose <Configured chargingProfilePurpose> AND - chargingProfile.stackLevel must be omitted AND - chargingProfile.chargingLimitSource must be omitted AND - chargingProfile.chargingProfileId must be omitted
Post scenario validations:
- N/a

TC_K_36_CSMS: Get Charging Profile - Evseld > 0 + chargingProfilePurpose + stackLevel

Test case name	Get Charging Profile - Evseld > 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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Test System responds with a GetChargingProfilesResponse with - status <i>Accepted</i>	1. The CSMS sends a GetChargingProfilesRequest
3. The Test System sends a ReportChargingProfilesRequest with - requestId <i>Generated Id</i>	4. The CSMS responds with a ReportChargingProfilesResponse

Tool validations
<p>* Step 1:</p> <p>Message GetChargingProfilesRequest</p> <ul style="list-style-type: none"> - evseld <Configured evseld> AND - chargingProfile.chargingProfilePurpose <Configured chargingProfilePurpose> AND - chargingProfile.chargingLimitSource must be omitted AND - chargingProfile.stackLevel <Configured stackLevel> AND - chargingProfile.chargingProfileId must be omitted
<p>Post scenario validations:</p> <ul style="list-style-type: none"> - N/a

TC_K_60_CSMS: Set Charging Profile - TxProfile with ongoing transaction on the specified EVSE

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 <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
2. The Test System responds with a SetChargingProfileResponse With status is <i>Accepted</i>	1. The CSMS sends a SetChargingProfileRequest

Tool validations
<p>* Step 1: (Message: SetChargingProfileRequest) chargingProfile.chargingProfilePurpose is <i>TxProfile</i> AND chargingProfile.evseId is <i><Configured evseId></i> AND chargingProfile.transactionId <i><Generated transactionId></i> AND chargingProfile.chargingProfileKind is <i>Relative</i> OR <i>Absolute</i> If chargingProfileKind is <i>Relative</i> then chargingSchedule.startSchedule must be omitted. If chargingProfileKind is <i>Absolute</i> then chargingSchedule.startSchedule must NOT be omitted.</p> <p>The received Charging Profile must comply with the requirements defined at part 2 specification.</p>
Post scenario validations: N/a

TC_K_37_CSMS: Remote start transaction with charging profile - Success

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 Test System responds with a RequestStartTransactionResponse With status <i>Accepted</i>	1. The CSMS sends a RequestStartTransactionRequest
3. The Test System sends a TransactionEventRequest With triggerReason <i>RemoteStart</i> transactionInfo.remoteStartId is present.	4. The CSMS responds with a TransactionEventResponse

Tool validations
<p>* Step 1:</p> <p>Message: RequestStartTransactionRequest</p> <p>idToken.idToken <Configured valid idToken></p> <p>idToken.type <Configured valid idToken type></p> <p>evseld <Configured evseld></p> <p>chargingProfile contains:</p> <p>chargingProfile.chargingProfilePurpose is <i>TxProfile</i></p> <p>chargingProfile.transactionId is omitted</p> <p>chargingProfile.chargingProfileKind is <i>Relative</i> OR <i>Absolute</i></p> <p>If chargingProfileKind is <i>Relative</i> then chargingSchedule.startSchedule must be omitted.</p> <p>If chargingProfileKind is <i>Absolute</i> then chargingSchedule.startSchedule must NOT be omitted.</p> <p>The received Charging Profile must comply with the requirements defined at part 2 specification.</p>
Post scenario validations: N/a

TC_K_43_CSMS: Get Composite Schedule - Specific EVSE

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
<p>2. The Test System 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</p>	<p>1. The CSMS sends a GetCompositeScheduleRequest</p>

Tool validations
<p>* Step 1: (Message: GetCompositeScheduleRequest) evseId 1 duration is <Configured duration> chargingRateUnit <Configured chargingRateUnit></p>
<p>Post scenario validations: N/a</p>

TC_K_44_CSMS: Get Composite Schedule - Charging Station

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
<p>2. The Test System responds with a GetCompositeScheduleResponse With status <i>Accepted</i> schedule.evseId 0 schedule.duration is 300 schedule.chargingRateUnit <Specified <i>chargingRateUnit</i> from step 1> schedule.chargingSchedulePeriod[0].startPeriod 0 <i>Note: Multiply limit by 1000 if chargingRateUnit is W</i> schedule.chargingSchedulePeriod[0].limit 10</p>	<p>1. The CSMS sends a GetCompositeScheduleRequest</p>

Tool validations
<p>* Step 1: (Message: GetCompositeScheduleRequest) evseId 0 duration is <Configured duration> chargingRateUnit <Configured chargingRateUnit></p>
<p>Post scenario validations: N/a</p>

TC_K_48_CSMS: EMS Control - Set / Update External Charging Limit (not on a transaction)

Test case name	EMS Control - 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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
1. The Test System sends a NotifyChargingLimitRequest with - chargingLimit.chargingLimitSource <i>EMS</i>	2. The CSMS responds with a NotifyChargingLimitResponse

Tool validations
- N/a
Post scenario validations: - N/a

TC_K_50_CSMS: EMS Control - Reset / release external charging limit - Without ongoing transaction

Test case name	EMS Control - 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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
1. The Test System sends a ClearedChargingLimitRequest with - chargingLimitSource <i>EMS</i>	2. The CSMS responds with a ClearedChargingLimitResponse

Tool validations
- N/a
Post scenario validations: - N/a

TC_K_51_CSMS: EMS Control - Reset / release external charging limit - With ongoing transaction

Test case name	EMS Control - 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 <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
1. The Test System sends a ClearedChargingLimitRequest with - chargingLimitSource <i>EMS</i>	2. The CSMS responds with a ClearedChargingLimitResponse
3. The Test System sends a TransactionEventRequest with - eventType <i>Updated</i> - triggerReason <i>ChargingRateChanged</i>	4. The CSMS responds with a TransactionEventResponse

Tool validations
- N/a
Post scenario validations: - N/a

TC_K_52_CSMS: EMS Control - Set / Update External Charging Limit (not on a transaction) - ChargingStationExternalConstraints in report

Test case name	EMS Control - 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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Test System responds with a GetChargingProfilesResponse with - status <i>Accepted</i>	1. The CSMS sends a GetChargingProfilesRequest
3. The Test System sends a ReportChargingProfilesRequest with - requestId <i>Generated Id</i> - chargingProfile.chargingProfilePurpose <i>ChargingStationExternalConstraints</i>	4. The CSMS responds with a ReportChargingProfilesResponse

Tool validations
N/a
Post scenario validations: - N/a

TC_K_53_CSMS: Charging with load leveling based on High Level Communication - Success

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 <i>Authorized</i> State is <i>EVConnectedPreSession</i>

Main (Test scenario)	
Charging Station	CSMS
1. Execute reusable state <i>ISO15118SmartCharging</i>	
2. The CSMS does NOT send a SetChargingProfileRequest	
<u>Note(s):</u> - <i>The CSMS must NOT initiate a renegotiate after starting the transaction, without cause. For example; a smart charging algorithm or an external trigger, etc.</i>	

Tool validations
- N/a
Post scenario validations: N/a

TC_K_55_CSMS: Charging with load leveling based on High Level Communication - EV charging profile exceeds limits

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 <i>Authorized</i> State is <i>EVConnectedPreSession</i>

Main (Test scenario)	
Charging Station	CSMS
1. The Test System sends a NotifyEVChargingNeedsRequest . With evseld <Configured evseld> maxScheduleTuples & chargingNeeds <Configured values from mock EV>	2. The CSMS responds with a NotifyEVChargingNeedsResponse .
4. The Test System responds with a SetChargingProfileResponse With status <i>Accepted</i>	3. The CSMS sends a SetChargingProfileRequest
5. The Test System sends a NotifyEVChargingScheduleRequest . With evseld <Configured evseld> chargingSchedule <ChargingSchedule that exceeds the limits of the chargingSchedule provided at step 3.>	6. The CSMS responds with a NotifyEVChargingScheduleResponse .
7. The Test System sends a TransactionEventRequest . With triggerReason <i>ChargingStateChanged</i> transactionInfo.chargingState <i>Charging</i>	8. The CSMS responds with a TransactionEventResponse .
10. The Test System responds with a SetChargingProfileResponse With status <i>Accepted</i>	9. The CSMS sends a SetChargingProfileRequest
11. The Test System sends a NotifyEVChargingScheduleRequest . With evseld <Configured evseld> chargingSchedule <ChargingSchedule provided at step 9>	12. The CSMS responds with a NotifyEVChargingScheduleResponse .

Tool validations
<p>* Step 2: (Message: NotifyEVChargingNeedsResponse) status <i>Accepted</i></p> <p>* Step 3: (Message: SetChargingProfileRequest) evseld <i><Configured evseld></i> chargingProfilePurpose <i>TxProfile</i> transactionId <i><Provided transactionId from before></i></p> <p>* Step 6: (Message: NotifyEVChargingScheduleResponse) status <i>Rejected</i></p> <p>* Step 9: (Message: SetChargingProfileRequest) evseld <i><Configured evseld></i> chargingProfilePurpose <i>TxProfile</i> transactionId <i><Provided transactionId from before></i></p> <p>* Step 12: (Message: NotifyEVChargingScheduleResponse) status <i>Accepted</i></p>
<p>Post scenario validations: N/a</p>

TC_K_57_CSMS: Renegotiating a Charging Schedule ISO 15118-20 - Initiated by EV

Test case name	Renegotiating a Charging Schedule ISO 15118-20 - 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 Test System sends a NotifyEVChargingNeedsRequest . With evseld <Configured evseld> maxScheduleTuples & chargingNeeds <Configured values from mock EV>	2. The CSMS responds with a NotifyEVChargingNeedsResponse .
4. The Test System responds with a SetChargingProfileResponse With status Accepted	3. The CSMS sends a SetChargingProfileRequest <u>Note(s):</u> - If NotifyEVChargingNeedsResponseStatus was Processing, the Test System will wait 60 seconds for the request
5. The Test System sends a NotifyEVChargingScheduleRequest . With evseld <Configured evseld> chargingSchedule <ChargingSchedule provided at step 3>	6. The CSMS responds with a NotifyEVChargingScheduleResponse .

Tool validations

* Step 2:

(Message: **NotifyEVChargingNeedsResponse**)**status** Accepted or Processing

* Step 3:

(Message: **SetChargingProfileRequest**)**evseld** <Configured evseld>**chargingProfilePurpose** TxProfile**transactionId** <Provided transactionId from before>

* Step 6:

(Message: **NotifyEVChargingScheduleResponse**)**status** Accepted

Post scenario validations:

N/a

TC_K_58_CSMS: Renegotiating a Charging Schedule ISO 15118-20 - Initiated by CSMS

Test case name	Renegotiating a Charging Schedule ISO 15118-2 - 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 <i>Authorized</i> AND <i>EVConnectedPreSession</i> AND <i>ISO15118SmartCharging</i>

Main (Test scenario)	
Charging Station	CSMS
2. The Test System responds with a SetChargingProfileResponse With status <i>Accepted</i>	1. The CSMS sends a SetChargingProfileRequest
3. The Test System sends a NotifyEVChargingScheduleRequest . With evseld <i><Configured evseld></i> chargingSchedule <i><ChargingSchedule provided at step 1></i>	4. The CSMS responds with a NotifyEVChargingScheduleResponse .

Tool validations
* Step 1: (Message: SetChargingProfileRequest) evseld <i><Configured evseld></i> chargingProfilePurpose <i>TxProfile</i> transactionId <i><Provided transactionId from before></i> * Step 4: (Message: NotifyEVChargingScheduleResponse) status <i>Accepted</i>
Post scenario validations: N/a

TC_K_59_CSMS: Renegotiating a Charging Schedule ISO 15118-20 - Initiated by CSMS - Send NotifyEVChargingNeeds

Test case name	Renegotiating a Charging Schedule ISO 15118-2 - 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 <i>Authorized</i> AND <i>EVConnectedPreSession</i> AND <i>ISO15118SmartCharging</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action:</u> <i>Trigger SetChargingProfile</i> evseld <Configured evseld> chargingProfilePurpose TxProfile transactionId <Provided transactionId from before>	
2. The Test System responds with a SetChargingProfileResponse With status Accepted	1. The CSMS sends a SetChargingProfileRequest
3. The Test System sends a NotifyEVChargingNeedsRequest . With evseld <Configured evseld> maxScheduleTuples & chargingNeeds <Configured values from mock EV>	4. The CSMS responds with a NotifyEVChargingNeedsResponse .
6. The Test System responds with a SetChargingProfileResponse With status Accepted	5. The CSMS sends a SetChargingProfileRequest <u>Note(s)</u> : - If NotifyEVChargingNeedsResponseStatus was Processing, the Test System will wait 60 seconds for the request
7. The Test System sends a NotifyEVChargingScheduleRequest . With evseld <Configured evseld> chargingSchedule <ChargingSchedule provided at step 3>	8. The CSMS responds with a NotifyEVChargingScheduleResponse .

Tool validations
<p>* Step 1: (Message: SetChargingProfileRequest) evseld <Configured evseld> chargingProfilePurpose TxProfile transactionId <Provided transactionId from before></p> <p>* Step 4: (Message: NotifyEVChargingNeedsResponse) status Accepted or Processing</p> <p>* Step 5: (Message: SetChargingProfileRequest) evseld <Configured evseld> chargingProfilePurpose TxProfile transactionId <Provided transactionId from before></p> <p>* Step 8: (Message: NotifyEVChargingScheduleResponse) status Accepted</p>
<p>Post scenario validations: N/a</p>

TC_K_70_CSMS: Set Charging Profile - Multiple Profiles

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 multiple Charging Profiles.
Prerequisite(s)	n/a

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State: State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
2. The Test System responds with a SetChargingProfileResponse with status <i>Accepted</i>	1. The CSMS sends a SetChargingProfileRequest with chargingProfilePurpose <i>TxDefaultProfile</i>
4. The Test System responds with a SetChargingProfileResponse with status <i>Accepted</i>	3. The CSMS sends a SetChargingProfileRequest with chargingProfilePurpose <i>ChargingStationMaxProfile</i>

Tool validations
<p>* Step 1: Message SetChargingProfileRequest chargingProfile.chargingProfilePurpose <i>TxDefaultProfile</i> chargingProfile.chargingProfileKind is <i>Absolute</i> OR <i>Recurring</i> chargingProfile.chargingSchedule.startSchedule must NOT be omitted. If chargingProfile.chargingProfileKind is <i>Recurring</i> then chargingProfile.recurrencyKind must NOT be omitted.</p> <p>* Step 3: Message SetChargingProfileRequest chargingProfile.id <different id from chargingProfile at step 1> chargingProfile.chargingProfilePurpose <i>ChargingStationMaxProfile</i> chargingProfile.chargingProfileKind is <i>Absolute</i> chargingProfile.chargingSchedule.startSchedule must NOT be omitted.</p> <p>The received Charging Profiles must comply with the requirements defined at part 2 specification.</p>
Post scenario validations: - N/a

TC_K_100_CSMS: Set Charging Profile - maxOfflineDuration

Test case name	Set Charging Profile - maxOfflineDuration
Test case Id	TC_K_100_CSMS
Use case Id(s)	K01
Requirement(s)	
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.
Purpose	To verify if the CSMS is able to specify a maxOfflineDuration and invalidAfterOfflineDuration the charging profile.
Prerequisite(s)	N/a

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
Manual Action: Request the CSMS to send a TxDefaultProfile charging profile with a maxOfflineDuration and invalidAfterOfflineDuration set to true	
2. The Test System responds with a SetChargingProfileResponse with status Accepted	1. The CSMS sends a SetChargingProfileRequest

Tool validations
* Step 1: Message SetChargingProfileRequest - chargingProfile.chargingProfilePurpose must be TxDefaultProfile - chargingProfile.maxOfflineDuration must be <not omitted> - chargingProfile.invalidAfterOfflineDuration must be true
Post scenario validations: N/a

TC_K_101_CSMS: Set Charging Profile - Change operation mode

Test case name	Set Charging Profile - Change operation mode
Test case Id	TC_K_101_CSMS
Use case Id(s)	K01
Requirement(s)	...
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.
Purpose	To verify if the CSMS is able to handle the message that the Charging Station sends when a charging profile changes the <code>operationMode</code> of a transaction.
Prerequisite(s)	N/a

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State: State is <i>ISO15118-20 transaction started in V2X AC_BPT energy transfer mode</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action:</u> Request the CSMS to send a TxDefaultProfile charging profile with operationMode Idle	
2. The Test System responds with a SetChargingProfileResponse with status <i>Accepted</i>	1. The CSMS sends a SetChargingProfileRequest
3. The Test System sends a TransactionEventRequest with eventType <i>_Updated</i> triggerReason <i>OperationModeChanged</i> transactionInfo.operationMode <i>Idle</i>	4. The CSMS responds with a TransactionEventResponse

Tool validations
<p>* Step 1: Message SetChargingProfileRequest - chargingProfile.chargingProfilePurpose must be <i>TxDefaultProfile</i> - chargingProfile.chargingSchedule.chargingSchedulePeriod[0].operationMode must be <i>Idle</i></p> <p>* Step 4: Message: TransactionEventResponse - status must be <i>Accepted</i></p>
Post scenario validations: N/a

TC_K_102_CSMS: Set Charging Profile - limitAtSoc

Test case name	Set Charging Profile - limitAtSoc
Test case Id	TC_K_102_CSMS
Use case Id(s)	K01
Requirement(s)	K01.FR.104
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.
Purpose	To verify if the CSMS is able to set a SoC limit for the transaction using a Charging Profile.
Prerequisite(s)	N/a

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action:</u> Request the CSMS to send a TxDefaultProfile charging profile with limitAtSoc specified	
2. The Test System responds with a SetChargingProfileResponse with status Accepted	1. The CSMS sends a SetChargingProfileRequest

Tool validations
* Step 1: Message SetChargingProfileRequest - chargingProfile.chargingProfilePurpose must be TxDefaultProfile - chargingProfile.chargingSchedule.limitAtSoC must be <not omitted>
Post scenario validations: N/a

TC_K_104_CSMS: Set Charging Profile - PriorityCharging

Test case name	Set Charging Profile - PriorityCharging
Test case Id	TC_K_104_CSMS
Use case Id(s)	K01
Requirement(s)	K01.FR.72, K01.FR.73
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.
Purpose	To verify if the CSMS is able to set a PriorityCharging Charging Profile.
Prerequisite(s)	N/a

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action:</u> Request the CSMS to send a PriorityCharging charging profile	
2. The Test System responds with a SetChargingProfileResponse with status Accepted	1. The CSMS sends a SetChargingProfileRequest

Tool validations
* Step 1: Message SetChargingProfileRequest - chargingProfile.chargingProfilePurpose must be <i>PriorityCharging</i> - chargingProfile.chargingSchedule.duration must be <omitted>
Post scenario validations: N/a

TC_K_106_CSMS: Set Charging Profile - randomizedDelay

Test case name	Set Charging Profile - randomizedDelay
Test case Id	TC_K_106_CSMS
Use case Id(s)	K01
Requirement(s)	K01.FR.96
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.
Purpose	To verify if the CSMS is able to set a Charging Profile with a randomizedDelay which uses useLocalTime.
Prerequisite(s)	N/a

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action:</u> Request the CSMS to send a TxDefaultProfile charging profile with useLocalTime true and a randomizedDelay > 0	
2. The Test System responds with a SetChargingProfileResponse with status Accepted	1. The CSMS sends a SetChargingProfileRequest

Tool validations
<p>* Step 1:</p> <p>Message SetChargingProfileRequest</p> <ul style="list-style-type: none"> - chargingProfile.chargingProfilePurpose must be TxDefaultProfile - chargingProfile.chargingSchedule.useLocalTime must be true - chargingProfile.chargingSchedule.randomizedDelay must be > 0
Post scenario validations: N/a

TC_K_109_CSMS: EMS Control - Set Charging Profile - MaxExternalConstraintsId

Test case name	EMS Control - Set Charging Profile - MaxExternalConstraintsId
Test case Id	TC_K_109_CSMS
Use case Id(s)	K01
Requirement(s)	K01.FR.80
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 charging profile using the right id set by MaxExternalConstraintsId as described at the OCPP specification.
Prerequisite(s)	n/a

Before (Preparations)

Configuration State:

N/a

Memory State:

N/a

Reusable State:

N/a

Main (Test scenario)

Charging Station	CSMS
2. The Test System responds with a GetBaseReportResponse with status <i>Accepted</i>	1. The CSMS sends a GetBaseReportRequest
3. The Test System sends a NotifyReportRequest containing requestId <requestId> reportData[0].component.name <i>SmartChargingCtrlr</i> reportData[0].variable.name <i>MaxExternalConstraintsId</i> reportData[0].variableAttribute.value <i>2147400000</i>	4. The CSMS responds with a NotifyReportResponse
<u>Manual Action:</u> Request the CSMS to send a TxDefaultProfile Charging Profile	
6. The Test System responds with a SetChargingProfileResponse with status <i>Accepted</i>	5. The CSMS sends a SetChargingProfileRequest

Tool validations

* Step 5:

Message **SetChargingProfileRequest****chargingProfile.id** must be greater than *2147400000*

Post scenario validations:

- N/a

TC_K_113_CSMS: Renegotiating a Charging Schedule ISO 15118-20 - Initiated by CSMS -

Test case name	Renegotiating a Charging Schedule ISO 15118-20 - Initiated by CSMS
Test case Id	TC_K_113_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)	CSMS supports ISO 15118-20

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>Authorized</i> State is <i>EVConnectedPreSession</i> State is <i>ISO15118SmartCharging</i> (iso15118-20)

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action:</u> <i>Trigger SetChargingProfile</i> evseld <Configured evseld> chargingProfilePurpose TxProfile transactionId <Provided transactionId from before>	
2. The Test System responds with a SetChargingProfileResponse With status <i>Accepted</i>	1. The CSMS sends a SetChargingProfileRequest
3. The Test System sends a NotifyEVChargingScheduleRequest . With evseld <Configured evseld> chargingSchedule <ChargingSchedule provided at step 1>	4. The CSMS responds with a NotifyEVChargingScheduleResponse .

Tool validations
* Step 1: (Message: SetChargingProfileRequest) evseld <Configured evseld> chargingProfilePurpose TxProfile transactionId <Provided transactionId from before> * Step 4: (Message: NotifyEVChargingScheduleResponse) status <i>Accepted</i>
Post scenario validations: N/a

TC_K_114_CSMS: Renegotiating a Charging Schedule ISO 15118-20 - Initiated by EV

Test case name	Renegotiating a Charging Schedule ISO 15118-20 - Initiated by EV
Test case Id	TC_K_114_CSMS
Use case Id(s)	K17
Requirement(s)	K17.FR.02, K17.FR.03, K17.FR.05, K17.FR.07, K17.FR.08, 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 <i>EVConnectedPreSession</i> State is <i>Authorized</i> State is <i>ISO15118SmartCharging</i> (iso15118-20)

Main (Test scenario)	
Charging Station	CSMS
<p>1. The Test System sends a NotifyEVChargingNeedsRequest with</p> <p>evseld <Configured evseld></p> <p>maxScheduleTuples 1</p> <p>chargingNeeds.requestedEnergyTransfer AC_three_phase</p> <p>chargingNeeds.availableEnergyTransfer [AC_single_phase, AC_three_phase, AC_BPT] chargingNeeds.controlMode ScheduledControl</p> <p>chargingNeeds.v2xChargingParameters.maxChargePower 6000</p> <p>chargingNeeds.v2xChargingParameters.maxDischargePower 0</p>	<p>2. The CSMS responds with a NotifyEVChargingNeedsResponse.</p>
<p>4. The Test System responds with a SetChargingProfileResponse with</p> <p>status Accepted</p>	<p>3. The CSMS sends a SetChargingProfileRequest</p> <p><u>Note(s):</u></p> <p>- If NotifyEVChargingNeedsResponseStatus was Processing, the Test System will wait 60 seconds for the request</p>
<p>5. The Test System sends a NotifyEVChargingScheduleRequest with</p> <p>evseld <Configured evseld></p> <p>chargingSchedule <ChargingSchedule provided at step 3></p>	<p>6. The CSMS responds with a NotifyEVChargingScheduleResponse.</p>

Tool validations
<div>* Step 2: Message: NotifyEVChargingNeedsResponse - status must be <i>Accepted</i> or <i>Processing</i></div> <div>* Step 3: Message: SetChargingProfileRequest - evseld must be <i><Configured evseld></i> - chargingProfilePurpose must be <i>TxProfile</i> - transactionId must be <i><Provided transactionId from before></i> - chargingProfile.chargingSchedule must be size = 1</div> <div>* Step 6: Message: NotifyEVChargingScheduleResponse - status must be <i>Accepted</i></div>
<div>Post scenario validations: N/a</div>

TC_K_115_CSMS: ISO 15118-20 Dynamic Control Mode - Success

Test case name	ISO 15118-20 Dynamic Control Mode - Success
Test case Id	TC_K_115_CSMS
Use case Id(s)	K17,K19
Requirement(s)	K17.FR.02, K17.FR.03, K17.FR.05, K17.FR.07, K17.FR.08, K17.FR.11, K17.FR.19, K19.FR.02, K19.FR.03, K19.FR.05, K19.FR.07, K19.FR.08
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 for DynamicControl and it provides new charging needs, which the Charging Station sends to the CSMS.
Prerequisite(s)	CSMS supports ISO 15118-20.

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 Test System sends a NotifyEVChargingNeedsRequest with evseld <Configured evseld> maxScheduleTuples 3 chargingNeeds.requestedEnergyTransfer <i>AC_BPT</i> chargingNeeds.availableEnergyTransfer [<i>AC_single_phase</i> , <i>AC_three_phase</i> , <i>AC_BPT</i>] chargingNeeds.controlMode <i>DynamicControl</i> chargingNeeds.v2xChargingParameters.maxChargePower 5000 chargingNeeds.v2xChargingParameters.maxDischargePower 5000	2. The CSMS responds with a NotifyEVChargingNeedsResponse .
4. The Test System responds with a SetChargingProfileResponse with status <i>Accepted</i>	3. The CSMS sends a SetChargingProfileRequest <u>Note(s):</u> - If NotifyEVChargingNeedsResponseStatus was <i>Processing</i> , the Test System will wait 60 seconds for the request
5. The Test System sends a NotifyEVChargingScheduleRequest with evseld <Configured evseld> chargingSchedule <ChargingSchedule provided at step 3>	6. The CSMS responds with a NotifyEVChargingScheduleResponse .

Tool validations
<p>* Step 2:</p> <p>Message: NotifyEVChargingNeedsResponse</p> <ul style="list-style-type: none">- status must be <i>Accepted</i> or <i>Processing</i> <p>* Step 3:</p> <p>Message: SetChargingProfileRequest</p> <ul style="list-style-type: none">- evseld must be <i><Configured evseld></i>- chargingProfilePurpose must be <i>TxProfile</i>- transactionId must be <i><Provided transactionId from before></i>- chargingProfile.chargingSchedule[*].chargingSchedulePeriods must be size ≤ 3- chargingProfile.chargingSchedule[*].absolutePriceSchedule.priceRuleStacks must be size ≤ 3- chargingProfile.chargingSchedule[*].priceLevelSchedule.priceLevelScheduleEntries must be size ≤ 3 <p>* Step 6:</p> <p>Message: NotifyEVChargingScheduleResponse</p> <ul style="list-style-type: none">- status must be <i>Accepted</i>
<p>Post scenario validations:</p> <p>N/a</p>

TC_K_117_CSMS: ISO 15118-20 Dynamic Control Mode - Adjusting charging schedule

Test case name	ISO 15118-20 Dynamic Control Mode - Adjusting charging schedule
Test case Id	TC_K_117_CSMS
Use case Id(s)	K20
Requirement(s)	K20.FR.03
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 provide a new charging profile when EV provides a changed departure time or target energy amount.
Prerequisite(s)	CSMS supports ISO15118-20

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>Authorized</i> State is <i>EVConnectedPreSession</i> State is <i>ISO15118SmartCharging</i> (iso15118-20)

Main (Test scenario)	
Charging Station	CSMS
<p>1. The Test System sends a NotifyEVChargingNeedsRequest with</p> <p>evseld <Configured evseld></p> <p>maxScheduleTuples 3</p> <p>chargingNeeds.requestedEnergyTransfer <i>AC_BPT</i></p> <p>chargingNeeds.availableEnergyTransfer [<i>AC_single_phase</i>, <i>AC_three_phase</i>, <i>AC_BPT</i>]</p> <p>chargingNeeds.controlMode <i>DynamicControl</i></p> <p>chargingNeeds.departureTime <Current DateTime + 2 hours></p> <p>chargingNeeds.v2xChargingParameters.evTargetEnergyRequest 1000</p> <p>chargingNeeds.v2xChargingParameters.maxChargePower 5000</p> <p>chargingNeeds.v2xChargingParameters.maxDischargePower 5000</p>	<p>2. The CSMS responds with a NotifyEVChargingNeedsResponse.</p>
<p>4. The Test System responds with a SetChargingProfileResponse with</p> <p>status <i>Accepted</i></p>	<p>3. The CSMS sends a SetChargingProfileRequest</p> <p><u>Note(s):</u> - If NotifyEVChargingNeedsResponseStatus was <i>Processing</i>, the Test System will wait 60 seconds for the request</p>
<p>5. The Test System sends a NotifyEVChargingScheduleRequest with</p> <p>evseld <Configured evseld></p> <p>chargingSchedule <ChargingSchedule provided at step 3></p>	<p>6. The CSMS responds with a NotifyEVChargingScheduleResponse.</p>

Main (Test scenario)	
<p>7. The Test System sends a NotifyEVChargingNeedsRequest with</p> <p>evseld <Configured evseld></p> <p>maxScheduleTuples 3</p> <p>chargingNeeds.requestedEnergyTransfer AC_BPT</p> <p>chargingNeeds.availableEnergyTransfer [AC_single_phase, AC_three_phase, AC_BPT]</p> <p>chargingNeeds.controlMode DynamicControl</p> <p>chargingNeeds.departureTime <Current DateTime + 3 hours></p> <p>chargingNeeds.v2xChargingParameters.evTargetEnergyRequest 1000</p> <p>chargingNeeds.v2xChargingParameters.maxChargePower 5000</p> <p>chargingNeeds.v2xChargingParameters.maxDischargePower 5000</p>	<p>8. The CSMS responds with a NotifyEVChargingNeedsResponse.</p>
<p>10. The Test System responds with a SetChargingProfileResponse with</p> <p>status Accepted</p>	<p>9. The CSMS sends a SetChargingProfileRequest</p> <p><u>Note(s):</u> - If NotifyEVChargingNeedsResponseStatus was Processing, the Test System will wait 60 seconds for the request</p>
<p>11. The Test System sends a NotifyEVChargingScheduleRequest with</p> <p>evseld <Configured evseld></p> <p>chargingSchedule <ChargingSchedule provided at step 5></p>	<p>12. The CSMS responds with a NotifyEVChargingScheduleResponse.</p>
<p>13. The Test System sends a NotifyEVChargingNeedsRequest with</p> <p>evseld <Configured evseld></p> <p>maxScheduleTuples 3</p> <p>chargingNeeds.requestedEnergyTransfer AC_BPT</p> <p>chargingNeeds.availableEnergyTransfer [AC_single_phase, AC_three_phase, AC_BPT]</p> <p>chargingNeeds.controlMode DynamicControl</p> <p>chargingNeeds.departureTime <Current DateTime + 3 hours></p> <p>chargingNeeds.v2xChargingParameters.evTargetEnergyRequest 3000</p> <p>chargingNeeds.v2xChargingParameters.maxChargePower 5000</p> <p>chargingNeeds.v2xChargingParameters.maxDischargePower 5000</p>	<p>14. The CSMS responds with a NotifyEVChargingNeedsResponse.</p>
<p>16. The Test System responds with a SetChargingProfileResponse with</p> <p>status Accepted</p>	<p>15. The CSMS sends a SetChargingProfileRequest</p> <p><u>Note(s):</u> - If NotifyEVChargingNeedsResponseStatus was Processing, the Test System will wait 60 seconds for the request</p>
<p>17. The Test System sends a NotifyEVChargingScheduleRequest with</p> <p>evseld <Configured evseld></p> <p>chargingSchedule <ChargingSchedule provided at step 5></p>	<p>18. The CSMS responds with a NotifyEVChargingScheduleResponse.</p>

Tool validations
<p>* Step 2:</p> <p>Message: NotifyEVChargingNeedsResponse</p> <ul style="list-style-type: none"> - status must be <i>Accepted</i> or <i>Processing</i> <p>* Step 3:</p> <p>Message: SetChargingProfileRequest</p> <ul style="list-style-type: none"> - evseld must be <i><Configured evseld></i> - chargingProfilePurpose must be <i>TxProfile</i> - transactionId must be <i><Provided transactionId from before></i> - chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].setpoint must be <i><Not omitted></i> <p>* Step 6:</p> <p>Message: NotifyEVChargingScheduleResponse</p> <ul style="list-style-type: none"> - status must be <i>Accepted</i> <p>* Step 8:</p> <p>Message: NotifyEVChargingNeedsResponse</p> <ul style="list-style-type: none"> - status must be <i>Accepted</i> or <i>Processing</i> <p>* Step 9:</p> <p>Message: SetChargingProfileRequest</p> <ul style="list-style-type: none"> - evseld must be <i><Configured evseld></i> - chargingProfilePurpose must be <i>TxProfile</i> - transactionId must be <i><Provided transactionId from before></i> <p>* Step 12:</p> <p>Message: NotifyEVChargingScheduleResponse</p> <ul style="list-style-type: none"> - status must be <i>Accepted</i> <p>* Step 14:</p> <p>Message: NotifyEVChargingNeedsResponse</p> <ul style="list-style-type: none"> - status must be <i>Accepted</i> or <i>Processing</i> <p>* Step 15:</p> <p>Message: SetChargingProfileRequest</p> <ul style="list-style-type: none"> - evseld must be <i><Configured evseld></i> - chargingProfilePurpose must be <i>TxProfile</i> - transactionId must be <i><Provided transactionId from before></i> <p>* Step 18:</p> <p>Message: NotifyEVChargingScheduleResponse</p> <ul style="list-style-type: none"> - status must be <i>Accepted</i>
<p>Post scenario validations:</p> <p>N/a</p>

TC_K_118_CSMS: Priority Charging - Requesting priority charging remotely

Test case name	Priority Charging - Requesting priority charging remotely
Test case Id	TC_K_118_CSMS
Use case Id(s)	K21
Requirement(s)	K21.FR.09, K21.FR.10
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 the CSMS supports priority charging initiated from CSMS.
Prerequisite(s)	N/a

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action:</u> Request the CSMS to use priority charging for the transaction	
2. The Test System responds with a UsePriorityChargingResponse with status <i>Accepted</i>	1. The CSMS sends a UsePriorityChargingRequest
3. The Test System sends a NotifyPriorityChargingRequest with transactionId <i><transactionId></i> activated <i>true</i>	4. The CSMS responds with a NotifyPriorityChargingResponse

Tool validations
* Step 1: Message: UsePriorityChargingRequest - transactionId must be <i><transactionId></i> - activate <i>true</i>
Post scenario validations: N/a

TC_K_121_CSMS: Dynamic charging profiles from CSMS - Pull

Test case name	Dynamic charging profiles from CSMS - Pull
Test case Id	TC_K_121_CSMS
Use case Id(s)	K28
Requirement(s)	K28.FR.01, K28.FR.02, K28.FR.07, K28.FR.12
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 support DynamicControl charging profiles.
Prerequisite(s)	N/a

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action</u> : Request the CSMS to send a dynamic charging profile to the CS	
2. The Test System responds with a SetChargingProfileResponse with status <i>Accepted</i>	1. The CSMS sends a SetChargingProfileRequest
3. The Test System sends a PullDynamicScheduleUpdateRequest with chargingProfileId < <i>unknown chargingProfileId</i> >	4. The CSMS responds with a PullDynamicScheduleUpdateResponse
5. The Test System sends a PullDynamicScheduleUpdateRequest with chargingProfileId must be < <i>chargingProfileId</i> received in step 1>	6. The CSMS responds with a PullDynamicScheduleUpdateResponse

Tool validations
<p>* Step 1: Message: SetChargingProfileRequest</p> <ul style="list-style-type: none"> - chargingProfile.chargingProfileKind must be <i>Dynamic</i> - chargingProfile.chargingSchedule must be size = 1 - chargingProfile.chargingSchedule[0].chargingSchedulePeriods must be size = 1 - chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].startPeriod must be 0 <p>* Step 4: Message: PullDynamicScheduleUpdateResponse</p> <ul style="list-style-type: none"> - status must be <i>Rejected</i> <p>* Step 6: Message: PullDynamicScheduleUpdateResponse</p> <ul style="list-style-type: none"> - status must be <i>Accepted</i> - scheduleUpdate must be <not omitted> - scheduleUpdate must at least have a single field set <p>Post scenario validations: N/a</p>

TC_K_126_CSMS: ISO 15118-20 Dynamic Control Mode - Sets no charging profile

Test case name	ISO 15118-20 Dynamic Control Mode - Sets no charging profile
Test case Id	TC_K_126_CSMS
Use case Id(s)	K19
Requirement(s)	K19.FR.02, K19.FR.04
System under test	CSMS
Description	Authorization of an EV by the CSMS to start a V2X power transfer loop.
Purpose	To check if the CSMS supports ISO15118-20 Dynamic Control without setting a charging profile.
Prerequisite(s)	CSMS supports ISO15118-20 The CSMS is able to turn off providing a Charging Profile during negotiation.

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is ISO15118-20 transaction just started exposing: - <allowedEnergyTransferModes> - <transactionId>

Main (Test scenario)	
Charging Station	CSMS
<p>1. The Test System sends a NotifyEVChargingNeedsRequest with:</p> <p>evseld <Configured evseld> chargingNeeds.requestedEnergyTransfer one of <allowedEnergyTransferModes> or if omitted AC_three_phase chargingNeeds.availableEnergyTransfer [AC_single_phase, AC_two_phase, AC_three_phase, AC_BPT, AC_BPT_DER, AC_DER] chargingNeeds.controlMode DynamicControl chargingNeeds.v2xChargingParameters.maxChargePower 8000W chargingNeeds.v2xChargingParameters.maxDischargePower 0W</p>	<p>2. Then CSMS SHALL respond with a NotifyEVChargingNeedsResponse</p>

Tool validations
* Step 2: Message: NotifyEVChargingNeedsResponse - status must be <i>NoChargingProfile</i>
Post scenario validations: N/a

L Firmware Management

TC_L_01_CSMS: Secure Firmware Update - Installation successful

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

Main (Test scenario)	
Charging Station	CSMS
2. The Test System responds with a UpdateFirmwareResponse With status <i>Accepted</i>	1. The CSMS sends a UpdateFirmwareRequest
3. The Test System sends a FirmwareStatusNotificationRequest . With status <i>Downloading</i>	4. The CSMS responds with a FirmwareStatusNotificationResponse .
5. The Test System sends a FirmwareStatusNotificationRequest . With status <i>Downloaded</i>	6. The CSMS responds with a FirmwareStatusNotificationResponse .
7. The Test System sends a FirmwareStatusNotificationRequest . With status <i>SignatureVerified</i>	8. The CSMS responds with a FirmwareStatusNotificationResponse .
9. The Test System sends a FirmwareStatusNotificationRequest . With status <i>Installing</i>	10. The CSMS responds with a FirmwareStatusNotificationResponse .
11. The Test System sends a FirmwareStatusNotificationRequest . With status <i>InstallRebooting</i>	12. The CSMS responds with a FirmwareStatusNotificationResponse .
13. The Test System sends a BootNotificationRequest With reason <i>FirmwareUpdate</i>	14. The CSMS responds with a BootNotificationResponse

Main (Test scenario)	
<p>15. The Test System notifies the CSMS about the current state of all connectors.</p> <p>Message: StatusNotificationRequest connectorStatus <i>Available</i> Message: NotifyEventRequest trigger <i>Delta</i> actualValue <i>"Available"</i> component.name <i>"Connector"</i> variable.name <i>"AvailabilityState"</i></p>	<p>16. The CSMS responds accordingly.</p>
<p>17. The Test System sends a FirmwareStatusNotificationRequest. With status <i>Installed</i></p>	<p>18. The CSMS responds with a FirmwareStatusNotificationResponse.</p>

Tool validations
<p>* Step 1: Message UpdateFirmwareRequest - firmware.signingCertificate <i><Configured signingCertificate></i> - firmware.signature <i><Configured signature></i></p> <p>* Step 14: Message BootNotificationResponse - status <i>Accepted</i></p>
<p>Post scenario validations: N/a</p>

TC_L_02_CSMS: Secure Firmware Update - InstallScheduled

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 <code>UpdateFirmwareRequest</code> with a <code>signingCertificate</code> .
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 <code>installDateTime</code> needs to be set to a future <code>dateTime</code> .

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

Main (Test scenario)	
Charging Station	CSMS
2. The Test System responds with a UpdateFirmwareResponse With status <i>Accepted</i>	1. The CSMS sends a UpdateFirmwareRequest
3. The Test System sends a FirmwareStatusNotificationRequest . With status <i>Downloading</i>	4. The CSMS responds with a FirmwareStatusNotificationResponse .
5. The Test System sends a FirmwareStatusNotificationRequest . With status <i>Downloaded</i>	6. The CSMS responds with a FirmwareStatusNotificationResponse .
7. The Test System sends a FirmwareStatusNotificationRequest . With status <i>SignatureVerified</i>	8. The CSMS responds with a FirmwareStatusNotificationResponse .
9. The Test System sends a FirmwareStatusNotificationRequest . With status <i>InstallScheduled</i>	10. The CSMS responds with a FirmwareStatusNotificationResponse .
11. The Test System sends a FirmwareStatusNotificationRequest . With status <i>Installing</i>	12. The CSMS responds with a FirmwareStatusNotificationResponse .
<u>Note(s):</u> - This step will be executed after the given <code>installDateTime</code> from step 1 has been reached.	
13. The Test System sends a FirmwareStatusNotificationRequest . With status <i>InstallRebooting</i>	14. The CSMS responds with a FirmwareStatusNotificationResponse .
15. The Test System sends a BootNotificationRequest With reason <i>FirmwareUpdate</i>	16. The CSMS responds with a BootNotificationResponse

Main (Test scenario)	
<p>17. The Test System notifies the CSMS about the current state of all connectors.</p> <p>Message: StatusNotificationRequest connectorStatus <i>Available</i> Message: NotifyEventRequest trigger <i>Delta</i> actualValue <i>"Available"</i> component.name <i>"Connector"</i> variable.name <i>"AvailabilityState"</i></p>	<p>18. The CSMS responds accordingly.</p>
<p>19. The Test System sends a FirmwareStatusNotificationRequest. With status <i>Installed</i></p>	<p>20. The CSMS responds with a FirmwareStatusNotificationResponse.</p>

Tool validations
<p>* Step 1: Message UpdateFirmwareRequest - firmware.installDateTime <i><A dateTime in the future></i></p> <p>* Step 16: Message BootNotificationResponse - status <i>Accepted</i></p>
<p>Post scenario validations: N/a</p>

TC_L_03_CSMS: Secure Firmware Update - DownloadScheduled

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 Test System responds with a UpdateFirmwareResponse With status <i>Accepted</i>	1. The CSMS sends a UpdateFirmwareRequest
3. The Test System sends a FirmwareStatusNotificationRequest . With status <i>DownloadScheduled</i>	4. The CSMS responds with a FirmwareStatusNotificationResponse .
5. The Test System sends a FirmwareStatusNotificationRequest . With status <i>Downloading</i> <u>Note(s):</u> - This step will be executed after the given retrieveDateTime from step 1 has been reached.	6. The CSMS responds with a FirmwareStatusNotificationResponse .
7. The Test System sends a FirmwareStatusNotificationRequest . With status <i>Downloaded</i>	8. The CSMS responds with a FirmwareStatusNotificationResponse .
9. The Test System sends a FirmwareStatusNotificationRequest . With status <i>SignatureVerified</i>	10. The CSMS responds with a FirmwareStatusNotificationResponse .
11. The Test System sends a FirmwareStatusNotificationRequest . With status <i>Installing</i>	12. The CSMS responds with a FirmwareStatusNotificationResponse .
13. The Test System sends a FirmwareStatusNotificationRequest . With status <i>InstallRebooting</i>	14. The CSMS responds with a FirmwareStatusNotificationResponse .
15. The Test System sends a BootNotificationRequest With reason <i>FirmwareUpdate</i>	16. The CSMS responds with a BootNotificationResponse

Main (Test scenario)	
<p>17. The Test System notifies the CSMS about the current state of all connectors.</p> <p>Message: StatusNotificationRequest connectorStatus <i>Available</i> Message: NotifyEventRequest trigger <i>Delta</i> actualValue <i>"Available"</i> component.name <i>"Connector"</i> variable.name <i>"AvailabilityState"</i></p>	<p>18. The CSMS responds accordingly.</p>
<p>19. The Test System sends a FirmwareStatusNotificationRequest. With status <i>Installed</i></p>	<p>20. The CSMS responds with a FirmwareStatusNotificationResponse.</p>

Tool validations
<p>* Step 1: Message UpdateFirmwareRequest - firmware.retrieveDateTime <A <i>dateTime</i> in the future></p> <p>* Step 16: Message BootNotificationResponse - status <i>Accepted</i></p>
<p>Post scenario validations: N/a</p>

TC_L_04_CSMS: Secure Firmware Update - RevokedCertificate

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 Test System responds with a UpdateFirmwareResponse With status <i>RevokedCertificate</i>	1. The CSMS sends a UpdateFirmwareRequest

Tool validations
N/a
Post scenario validations: N/a

TC_L_05_CSMS: Secure Firmware Update - InvalidCertificate

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 Test System responds with a UpdateFirmwareResponse With status <i>InvalidCertificate</i>	1. The CSMS sends a UpdateFirmwareRequest
3. The Test System sends a SecurityEventNotificationRequest With type is <i>InvalidFirmwareSigningCertificate</i>	4. The CSMS responds with a SecurityEventNotificationResponse

Tool validations
N/a
Post scenario validations: N/a

TC_L_06_CSMS: Secure Firmware Update - InvalidSignature

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 Test System responds with a UpdateFirmwareResponse With status <i>Accepted</i>	1. The CSMS sends a UpdateFirmwareRequest
3. The Test System sends a FirmwareStatusNotificationRequest . With status <i>Downloading</i>	4. The CSMS responds with a FirmwareStatusNotificationResponse .
5. The Test System sends a FirmwareStatusNotificationRequest . With status <i>Downloaded</i>	6. The CSMS responds with a FirmwareStatusNotificationResponse .
7. The Test System sends a FirmwareStatusNotificationRequest . With status <i>InvalidSignature</i>	8. The CSMS responds with a FirmwareStatusNotificationResponse .
9. The Test System sends a SecurityEventNotificationRequest With type is <i>InvalidFirmwareSignature</i>	10. The CSMS responds with a SecurityEventNotificationResponse

Tool validations
N/a
Post scenario validations: N/a

TC_L_07_CSMS: Secure Firmware Update - DownloadFailed

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 <code>UpdateFirmwareRequest</code> with a <code>signingCertificate</code> .
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 Test System responds with a UpdateFirmwareResponse With status <i>Accepted</i>	1. The CSMS sends a UpdateFirmwareRequest
3. The Test System sends a FirmwareStatusNotificationRequest . With status <i>Downloading</i>	4. The CSMS responds with a FirmwareStatusNotificationResponse .
5. The Test System sends a FirmwareStatusNotificationRequest . With status <i>DownloadFailed</i>	6. The CSMS responds with a FirmwareStatusNotificationResponse .

Tool validations
N/a
Post scenario validations: N/a

TC_L_08_CSMS: Secure Firmware Update - InstallVerificationFailed

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 Test System responds with a UpdateFirmwareResponse With status <i>Accepted</i>	1. The CSMS sends a UpdateFirmwareRequest
3. The Test System sends a FirmwareStatusNotificationRequest . With status <i>Downloading</i>	4. The CSMS responds with a FirmwareStatusNotificationResponse .
5. The Test System sends a FirmwareStatusNotificationRequest . With status <i>Downloaded</i>	6. The CSMS responds with a FirmwareStatusNotificationResponse .
7. The Test System sends a FirmwareStatusNotificationRequest . With status <i>SignatureVerified</i>	8. The CSMS responds with a FirmwareStatusNotificationResponse .
9. The Test System sends a FirmwareStatusNotificationRequest . With status <i>Installing</i>	10. The CSMS responds with a FirmwareStatusNotificationResponse .
11. The Test System sends a FirmwareStatusNotificationRequest . With status <i>InstallVerificationFailed</i>	12. The CSMS responds with a FirmwareStatusNotificationResponse .

Tool validations

N/a

Post scenario validations:

N/a

TC_L_09_CSMS: Secure Firmware Update - InstallationFailed

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 Test System responds with a UpdateFirmwareResponse With status <i>Accepted</i>	1. The CSMS sends a UpdateFirmwareRequest
3. The Test System sends a FirmwareStatusNotificationRequest . With status <i>Downloading</i>	4. The CSMS responds with a FirmwareStatusNotificationResponse .
5. The Test System sends a FirmwareStatusNotificationRequest . With status <i>Downloaded</i>	6. The CSMS responds with a FirmwareStatusNotificationResponse .
7. The Test System sends a FirmwareStatusNotificationRequest . With status <i>SignatureVerified</i>	8. The CSMS responds with a FirmwareStatusNotificationResponse .
9. The Test System sends a FirmwareStatusNotificationRequest . With status <i>Installing</i>	10. The CSMS responds with a FirmwareStatusNotificationResponse .
11. The Test System sends a FirmwareStatusNotificationRequest . With status <i>InstallRebooting</i>	12. The CSMS responds with a FirmwareStatusNotificationResponse .
13. The Test System sends a BootNotificationRequest With reason <i>FirmwareUpdate</i>	14. The CSMS responds with a BootNotificationResponse
15. The Test System notifies the CSMS about the current state of all connectors. Message: StatusNotificationRequest connectorStatus <i>Available</i> Message: NotifyEventRequest trigger <i>Delta</i> actualValue <i>"Available"</i> component.name <i>"Connector"</i> variable.name <i>"AvailabilityState"</i>	16. The CSMS responds accordingly.

Main (Test scenario)	
17. The Test System sends a FirmwareStatusNotificationRequest . With status <i>InstallationFailed</i>	18. The CSMS responds with a FirmwareStatusNotificationResponse .

Tool validations
* Step 14: Message BootNotificationResponse - status <i>Accepted</i>
Post scenario validations: N/a

TC_L_10_CSMS: Secure Firmware Update - AcceptedCanceled

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 Test System responds with a UpdateFirmwareResponse With status Accepted	1. The CSMS sends a UpdateFirmwareRequest
3. The Test System sends a FirmwareStatusNotificationRequest . With status Downloading	4. The CSMS responds with a FirmwareStatusNotificationResponse .
6. The Test System responds with a UpdateFirmwareResponse With status AcceptedCanceled	5. The CSMS sends a UpdateFirmwareRequest
7. The Test System sends a FirmwareStatusNotificationRequest . With status Downloading	8. The CSMS responds with a FirmwareStatusNotificationResponse .
9. The Test System sends a FirmwareStatusNotificationRequest . With status Downloaded	10. The CSMS responds with a FirmwareStatusNotificationResponse .
11. The Test System sends a FirmwareStatusNotificationRequest . With status SignatureVerified	12. The CSMS responds with a FirmwareStatusNotificationResponse .
13. The Test System sends a FirmwareStatusNotificationRequest . With status Installing	14. The CSMS responds with a FirmwareStatusNotificationResponse .
15. The Test System sends a FirmwareStatusNotificationRequest . With status InstallRebooting	16. The CSMS responds with a FirmwareStatusNotificationResponse .
17. The Test System sends a BootNotificationRequest With reason FirmwareUpdate	18. The CSMS responds with a BootNotificationResponse

Main (Test scenario)	
<p>19. The Test System notifies the CSMS about the current state of all connectors.</p> <p>Message: StatusNotificationRequest connectorStatus <i>Available</i> Message: NotifyEventRequest trigger <i>Delta</i> actualValue <i>"Available"</i> component.name <i>"Connector"</i> variable.name <i>"AvailabilityState"</i></p>	<p>20. The CSMS responds accordingly.</p>
<p>21. The Test System sends a FirmwareStatusNotificationRequest. With status <i>Installed</i></p>	<p>22. The CSMS responds with a FirmwareStatusNotificationResponse.</p>

Tool validations
<p>* Step 18: Message BootNotificationResponse - status <i>Accepted</i></p>
<p>Post scenario validations: N/a</p>

TC_L_11_CSMS: Secure Firmware Update - Unable to cancel

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 Test System responds with a UpdateFirmwareResponse With status Accepted	1. The CSMS sends a UpdateFirmwareRequest
3. The Test System sends a FirmwareStatusNotificationRequest . With status Downloading	4. The CSMS responds with a FirmwareStatusNotificationResponse .
6. The Test System responds with a UpdateFirmwareResponse With status Rejected	5. The CSMS sends a UpdateFirmwareRequest
7. The Test System sends a FirmwareStatusNotificationRequest . With status Downloaded	8. The CSMS responds with a FirmwareStatusNotificationResponse .
9. The Test System sends a FirmwareStatusNotificationRequest . With status SignatureVerified	10. The CSMS responds with a FirmwareStatusNotificationResponse .
11. The Test System sends a FirmwareStatusNotificationRequest . With status Installing	12. The CSMS responds with a FirmwareStatusNotificationResponse .
13. The Test System sends a FirmwareStatusNotificationRequest . With status InstallRebooting	14. The CSMS responds with a FirmwareStatusNotificationResponse .
15. The Test System sends a BootNotificationRequest With reason FirmwareUpdate	16. The CSMS responds with a BootNotificationResponse

Main (Test scenario)	
<p>17. The Test System notifies the CSMS about the current state of all connectors.</p> <p>Message: StatusNotificationRequest connectorStatus <i>Available</i> Message: NotifyEventRequest trigger <i>Delta</i> actualValue <i>"Available"</i> component.name <i>"Connector"</i> variable.name <i>"AvailabilityState"</i></p>	<p>18. The CSMS responds accordingly.</p>
<p>19. The Test System sends a FirmwareStatusNotificationRequest. With status <i>Installed</i></p>	<p>20. The CSMS responds with a FirmwareStatusNotificationResponse.</p>

Tool validations
<p>* Step 16: Message BootNotificationResponse - status <i>Accepted</i></p>
<p>Post scenario validations: N/a</p>

TC_L_13_CSMS: Secure Firmware Update - Unable to download/install firmware with ongoing transaction - AllowNewSessionsPendingFirmwareUpdate is false

Test case name	Secure Firmware Update - Unable to download/install firmware with ongoing transaction - AllowNewSessionsPendingFirmwareUpdate is false
Test case Id	TC_L_13_CSMS
Use case Id(s)	L01
Requirement(s)	L01.FR.01,L01.FR.11
System under test	CSMS
Description	The CSMS is able to request the Charging Station to securely download and install a new firmware by sending an UpdateFirmwareRequest with a signingCertificate.
Purpose	To verify if the CSMS is able to handle a Charging Station setting connectors to Unavailable while preparing a firmware update when there is a transaction ongoing.
Prerequisite(s)	The CSMS is able to request a new firmware update when there is a transaction ongoing on the Charging Station.

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
2. The Test System responds with a UpdateFirmwareResponse With status <i>Accepted</i>	1. The CSMS sends a UpdateFirmwareRequest
3. The Test System sends a FirmwareStatusNotificationRequest . With status <i>DownloadScheduled</i>	4. The CSMS responds with a FirmwareStatusNotificationResponse .
5. The Test System notifies the CSMS about the state change of all connectors that don't have a running transaction. Message: StatusNotificationRequest connectorStatus <i>Unavailable</i> Message: NotifyEventRequest trigger <i>Delta</i> actualValue <i>"Unavailable"</i> component.name <i>"Connector"</i> variable.name <i>"AvailabilityState"</i>	6. The CSMS responds accordingly.
7. Execute Reusable State <i>StopAuthorized</i> <u>Note(s)</u> Wait <configured transaction duration> before executing this step	
8. Execute Reusable State <i>EVConnectedPostSession</i>	
9. Execute Reusable State <i>EVDisconnected</i>	
10. The Test System sends a FirmwareStatusNotificationRequest . With status <i>Downloading</i> <u>Note(s)</u> : - This step will be executed after the given retrieveDateTime from step 1 has been reached.	11. The CSMS responds with a FirmwareStatusNotificationResponse .

Main (Test scenario)	
12. The Test System sends a FirmwareStatusNotificationRequest . With status <i>Downloaded</i>	13. The CSMS responds with a FirmwareStatusNotificationResponse .
14. The Test System sends a FirmwareStatusNotificationRequest . With status <i>SignatureVerified</i>	15. The CSMS responds with a FirmwareStatusNotificationResponse .
16. The Test System sends a FirmwareStatusNotificationRequest . With status <i>Installing</i>	17. The CSMS responds with a FirmwareStatusNotificationResponse .
18. The Test System sends a FirmwareStatusNotificationRequest . With status <i>InstallRebooting</i>	19. The CSMS responds with a FirmwareStatusNotificationResponse .
20. The Test System sends a BootNotificationRequest With reason <i>FirmwareUpdate</i>	21. The CSMS responds with a BootNotificationResponse
22. The Test System notifies the CSMS about the current state of all connectors. Message: StatusNotificationRequest connectorStatus <i>Available</i> Message: NotifyEventRequest trigger <i>Delta</i> actualValue <i>"Available"</i> component.name <i>"Connector"</i> variable.name <i>"AvailabilityState"</i>	23. The CSMS responds accordingly.
24. The Test System sends a FirmwareStatusNotificationRequest . With status <i>Installed</i>	25. The CSMS responds with a FirmwareStatusNotificationResponse .

Tool validations
* Step 1: Message UpdateFirmwareRequest - firmware.signingCertificate <i><configured signingCertificate></i> * Step 19: Message BootNotificationResponse - status <i>Accepted</i>
Post scenario validations: N/a

TC_L_17_CSMS: Publish Firmware - Published

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Test System responds with a PublishFirmwareResponse with status <i>Accepted</i>	1. The CSMS sends a PublishFirmwareRequest
3. The Test System sends a PublishFirmwareStatusNotificationRequest with status <i>Downloading</i>	4. The CSMS responds with a PublishFirmwareStatusNotificationResponse
5. The Test System sends a PublishFirmwareStatusNotificationRequest with status <i>Downloaded</i>	6. The CSMS responds with a PublishFirmwareStatusNotificationResponse
7. The Test System sends a PublishFirmwareStatusNotificationRequest with status <i>ChecksumVerified</i>	8. The CSMS responds with a PublishFirmwareStatusNotificationResponse
9. The Test System sends a PublishFirmwareStatusNotificationRequest with status <i>Published</i> AND location <i><Configured firmware_location></i>	10. The CSMS responds with a PublishFirmwareStatusNotificationResponse

Tool validations
* Step 1: Message PublishFirmwareRequest - location <i><Configured firmware_location></i>
Post scenario validations: - N/a

TC_L_24_CSMS: Publish Firmware - Download failed

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Test System responds with a PublishFirmwareResponse with status <i>Accepted</i>	1. The CSMS sends a PublishFirmwareRequest
3. The Test System sends a PublishFirmwareStatusNotificationRequest with status <i>Downloading</i>	4. The CSMS responds with a PublishFirmwareStatusNotificationResponse
5. The Test System sends a PublishFirmwareStatusNotificationRequest with status <i>DownloadFailed</i>	6. The CSMS responds with a PublishFirmwareStatusNotificationResponse

Tool validations
* Step 1: Message PublishFirmwareRequest - location <Configured firmware_location>
Post scenario validations: - N/a

TC_L_19_CSMS: Publish Firmware - Invalid Checksum

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Test System responds with a PublishFirmwareResponse with status <i>Accepted</i>	1. The CSMS sends a PublishFirmwareRequest
3. The Test System sends a PublishFirmwareStatusNotificationRequest with status <i>Downloading</i>	4. The CSMS responds with a PublishFirmwareStatusNotificationResponse
5. The Test System sends a PublishFirmwareStatusNotificationRequest with status <i>Downloaded</i>	6. The CSMS responds with a PublishFirmwareStatusNotificationResponse
7. The Test System sends a PublishFirmwareStatusNotificationRequest with status <i>InvalidChecksum</i>	8. The CSMS responds with a PublishFirmwareStatusNotificationResponse

Tool validations
* Step 1: Message PublishFirmwareRequest - location <Configured firmware_location>
Post scenario validations: - N/a

TC_L_20_CSMS: Publish Firmware - PublishFailed

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Test System responds with a PublishFirmwareResponse with status <i>Accepted</i>	1. The CSMS sends a PublishFirmwareRequest
3. The Test System sends a PublishFirmwareStatusNotificationRequest with status <i>Downloading</i>	4. The CSMS responds with a PublishFirmwareStatusNotificationResponse
5. The Test System sends a PublishFirmwareStatusNotificationRequest with status <i>Downloaded</i>	6. The CSMS responds with a PublishFirmwareStatusNotificationResponse
7. The Test System sends a PublishFirmwareStatusNotificationRequest with status <i>ChecksumVerified</i>	8. The CSMS responds with a PublishFirmwareStatusNotificationResponse
9. The Test System sends a PublishFirmwareStatusNotificationRequest with status <i>PublishFailed</i>	10. The CSMS responds with a PublishFirmwareStatusNotificationResponse

Tool validations
* Step 1: Message PublishFirmwareRequest - location <Configured firmware_location>
Post scenario validations: - N/a

TC_L_21_CSMS: Unpublish Firmware - Unpublished

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Test System responds with a UnpublishFirmwareResponse with status <i>Unpublished</i>	1. The CSMS sends a UnpublishFirmwareRequest

Tool validations
-N/a
Post scenario validations: - N/a

TC_L_22_CSMS: Unpublish Firmware - NoFirmware

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Test System responds with a UnpublishFirmwareResponse with status NoFirmware	1. The CSMS sends a UnpublishFirmwareRequest

Tool validations
-N/a
Post scenario validations: - N/a

TC_L_23_CSMS: Unpublish Firmware - Download Ongoing

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Test System responds with a UnpublishFirmwareResponse with status <i>DownloadOngoing</i>	1. The CSMS sends a UnpublishFirmwareRequest

Tool validations
-N/a
Post scenario validations: - N/a

M Certificate Management

TC_M_01_CSMS: Install CA certificate - CSMSRootCertificate

Test case name	Install CA certificate - CSMSRootCertificate
Test case Id	TC_M_01_CSMS
Use case Id(s)	M05
Requirement(s)	M05.FR.01
System under test	CSMS
Description	The CSMS is able to request the Charging Station to install new Root CA certificates using the InstallCertificateRequest message.
Purpose	To verify if the CSMS is able to request a Charging Station to install a new CSMSRootCertificate.
Prerequisite(s)	N/a

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

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>CertificateInstalled</i> for certificateType <i>CSMSRootCertificate</i>	

Tool validations
N.a
Post scenario validations: N/a

TC_M_02_CSMS: Install CA certificate - ManufacturerRootCertificate

Test case name	Install CA certificate - ManufacturerRootCertificate
Test case Id	TC_M_02_CSMS
Use case Id(s)	M05
Requirement(s)	M05.FR.01
System under test	CSMS
Description	The CSMS is able to request the Charging Station to install new Root CA certificates using the InstallCertificateRequest message.
Purpose	To verify if the CSMS is able to request a Charging Station to install a new ManufacturerRootCertificate.
Prerequisite(s)	N/a

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

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>CertificateInstalled</i> for certificateType <i>ManufacturerRootCertificate</i>	

Tool validations
N/a
Post scenario validations: N/a

TC_M_03_CSMS: Install CA certificate - V2GRootCertificate

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 <i>CertificateInstalled</i> for certificateType V2GRootCertificate	

Tool validations
N/a
Post scenario validations: N/a

TC_M_04_CSMS: Install CA certificate - MORootCertificate

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 <i>CertificateInstalled</i> for certificateType <i>MORootCertificate</i>	

Tool validations
N/a
Post scenario validations: N/a

TC_M_05_CSMS: Install CA certificate - Failed

Test case name	Install CA certificate - Failed
Test case Id	TC_M_05_CSMS
Use case Id(s)	M05
Requirement(s)	M05.FR.01,M05.FR.03
System under test	CSMS
Description	The CSMS is able to request the Charging Station to install new Root CA certificates using the InstallCertificateRequest message.
Purpose	To verify if the CSMS is able to handle a Charging Station reporting it failed to install the requested certificate.
Prerequisite(s)	N/a

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

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action:</u> Trigger the CSMS to send an InstallCertificateRequest with certificateType CSMSRootCertificate.	
2. The Test System responds with a InstallCertificateResponse With status is Failed	1. The CSMS sends a InstallCertificateRequest

Tool validations
* Step 1: Message: InstallCertificateRequest certificateType must be CSMSRootCertificate certificate contains <A certificate>
Post scenario validations: N/a

TC_M_12_CSMS: Retrieve certificates from Charging Station - CSMSRootCertificate

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 <i>GetInstalledCertificates</i> for certificateType <i>CSMSRootCertificate</i> . The Test System responds with data hashed with SHA256.	
2. Execute Reusable State <i>GetInstalledCertificates</i> for certificateType <i>CSMSRootCertificate</i> . The Test System responds with data hashed with SHA384.	
3. Execute Reusable State <i>GetInstalledCertificates</i> for certificateType <i>CSMSRootCertificate</i> . The Test System responds with data hashed with SHA512.	

Tool validations
N/a
Post scenario validations: N/a

TC_M_13_CSMS: Retrieve certificates from Charging Station - ManufacturerRootCertificate

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 <i>GetInstalledCertificates</i> for certificateType <i>ManufacturerRootCertificate</i>	

Tool validations
N/a
Post scenario validations: N/a

TC_M_14_CSMS: Retrieve certificates from Charging Station - V2GRootCertificate

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 <i>GetInstalledCertificates</i> for certificateType V2GRootCertificate	

Tool validations
N/a
Post scenario validations: N/a

TC_M_15_CSMS: Retrieve certificates from Charging Station - V2GCertificateChain

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 <i>GetInstalledCertificates</i> for certificateType V2GCertificateChain	

Tool validations
N/a
Post scenario validations: N/a

TC_M_16_CSMS: Retrieve certificates from Charging Station - MORootCertificate

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 <i>GetInstalledCertificates</i> for certificateType <i>MORootCertificate</i>	

Tool validations
N/a
Post scenario validations: N/a

TC_M_17_CSMS: Retrieve certificates from Charging Station - CSMSRootCertificate & ManufacturerRootCertificate

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 <i>GetInstalledCertificates</i> for certificateType CSMSRootCertificate AND ManufacturerRootCertificate	

Tool validations
N/a
Post scenario validations: N/a

TC_M_18_CSMS: Retrieve certificates from Charging Station - All certificateTypes

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 <code>GetInstalledCertificateIdsRequest</code> without <code>certificateType</code> .	
2. The Test System responds with a GetInstalledCertificateIdsResponse With status is <i>Accepted</i> certificateHashDataChain contains <i><The hashData of all certificates stored at the Test System truststore></i>	1. The CSMS sends a GetInstalledCertificateIdsRequest

Tool validations
* Step 1: Message: GetInstalledCertificateIdsRequest - certificateType is omitted
Post scenario validations: N/a

TC_M_19_CSMS: Retrieve certificates from Charging Station - No matching certificate found

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 <code>GetInstalledCertificateIdsRequest</code> message.
Purpose	To verify if the CSMS is able to handle a response from the Charging Station indicating it was not able to find a certificate for the requested criteria.
Prerequisite(s)	N/a

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

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action:</u> Trigger the CSMS to send a <code>GetInstalledCertificateIdsRequest</code> with <code>certificateType ManufacturerRootCertificate</code> .	
2. The Test System responds with a <code>GetInstalledCertificateIdsResponse</code> With status is <i>NotFound</i> certificateHashDataChain is omitted.	1. The CSMS sends a <code>GetInstalledCertificateIdsRequest</code>

Tool validations
* Step 1: Message: <code>GetInstalledCertificateIdsRequest</code> - certificateType is <i>ManufacturerRootCertificate</i>
Post scenario validations: N/a

TC_M_20_CSMS: Delete a certificate from a Charging Station - Success

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. <i>CertificateInstalled</i> with certificateType <i>CSMSRootCertificate</i> .	
<u>Manual Action</u> : Request the CSMS to send a <i>DeleteCertificateRequest</i> .	
3. The Test System responds with a GetInstalledCertificateIdsResponse With status is <i>Accepted</i> certificateHashDataChain contains an entry with following values: certificateHashDataChain[0].certificateType is <i>CSMSRootCertificate</i> certificateHashDataChain[0].certificateHashData.hashAlgorithm is <i>SHA256</i>	2. The CSMS sends a GetInstalledCertificateIdsRequest
5. The Test System responds with a DeleteCertificateResponse With status is <i>Accepted</i>	4. The CSMS sends a DeleteCertificateRequest
<u>Note(s)</u> : - Steps 1 - 5 will be repeated for each hash algorithm (<i>SHA256</i> , <i>SHA384</i> , <i>SHA512</i>).	

Tool validations
* Step 2: Message: GetInstalledCertificateIdsRequest - certificateType contains <i>CSMSRootCertificate</i> OR is <i>omitted</i> .
* Step 4: Message: DeleteCertificateRequest - certificateHashData is <i><Returned certificateHashData at Step 3></i> .
Post scenario validations: N/a

TC_M_21_CSMS: Delete a certificate from a Charging Station - Failed

Test case name	Delete a certificate from a Charging Station - Failed
Test case Id	TC_M_21_CSMS
Use case Id(s)	M04
Requirement(s)	M04.FR.01,M04.FR.07
System under test	CSMS
Description	The CSMS is able to request the Charging Station to delete an installed certificate using the DeleteCertificateRequest message.
Purpose	To verify if CSMS is able to handle a Charging Station that fails to delete an installed certificate.
Prerequisite(s)	N/a

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): <i>CertificateInstalled</i> with certificateType <i>CSMSRootCertificate</i> .

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action</u> : Request the CSMS to send a DeleteCertificateRequest.	
2. The Test System responds with a GetInstalledCertificateIdsResponse With status is <i>Accepted</i> certificateHashDataChain contains an entry with following values: certificateHashDataChain[0].certificateType is <i>CSMSRootCertificate</i> certificateHashDataChain[0].certificateHashData.hashAlgorithm is <i>SHA256</i>	1. The CSMS sends a GetInstalledCertificateIdsRequest
4. The Test System responds with a DeleteCertificateResponse With status is <i>Failed</i>	3. The CSMS sends a DeleteCertificateRequest

Tool validations
* Step 1: Message: GetInstalledCertificateIdsRequest - certificateType contains <i>CSMSRootCertificate</i> OR is <i>omitted</i> .
* Step 3: Message: DeleteCertificateRequest - certificateHashData contains <i><Returned certificateHashData at Step 2></i> .
Post scenario validations: N/a

TC_M_24_CSMS: Get Charging Station Certificate status - Success

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 Test System sends one or more subsequent GetCertificateStatusRequests With ocspRequestData contains <i><hashes from configured (V2G) certificate chain SubCA's></i>	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

TC_M_26_CSMS: Certificate Installation EV - Success

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 Test System sends a Get15118EVCertificateRequest With action Install	2. The CSMS responds with a Get15118EVCertificateResponse

Tool validations
* Step 2: Message: Get15118EVCertificateResponse - status <i>Accepted</i> - exiResponse <i><Raw CertificateInstallationRes response for the EV, Base64 encoded.></i>
Post scenario validations: N/a

TC_M_28_CSMS: Certificate Update EV - Success

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 Test System sends a Get15118EVCertificateRequest With action Update	2. The CSMS responds with a Get15118EVCertificateResponse

Tool validations
* Step 2: Message: Get15118EVCertificateResponse - status <i>Accepted</i> - exiResponse <i><Raw CertificateInstallationRes response for the EV, Base64 encoded.></i>
Post scenario validations: N/a

TC_M_100_CSMS: Certificate Installation EV - ISO 15118-20 - Success

Test case name	Certificate Installation EV - ISO 15118-20 - Success
Test case Id	TC_M_100_CSMS
Use case Id(s)	M01
Requirement(s)	M01.FR.02, M01.FR.05
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
<u>Note(s)</u> : 3 ContractCertificateChains will be communicated to the CSMS.	
1. The Test System sends a Get15118EVCertificateRequest With action <i>Install</i> maximumContractCertificateChains 10 prioritizedEMAIDs <Configured list EMA IDs>	2. The CSMS responds with a Get15118EVCertificateResponse
3. The Test System sends a Get15118EVCertificateRequest With action <i>Install</i> maximumContractCertificateChains 10 prioritizedEMAIDs <Configured list EMA IDs>	4. The CSMS responds with a Get15118EVCertificateResponse
5. The Test System sends a Get15118EVCertificateRequest With action <i>Install</i> maximumContractCertificateChains 10 prioritizedEMAIDs <Configured list EMA IDs>	6. The CSMS responds with a Get15118EVCertificateResponse

Tool validations
<p>* Step 2:</p> <p>Message: Get15118EVCertificateResponse</p> <ul style="list-style-type: none"> - status must be <i>Accepted</i> - exiResponse <Raw CertificateInstallationRes response for the EV, Base64 encoded.> - remainingContracts must be 2 <p>* Step 4:</p> <p>Message: Get15118EVCertificateResponse</p> <ul style="list-style-type: none"> - status must be <i>Accepted</i> - exiResponse <Raw CertificateInstallationRes response for the EV, Base64 encoded.> - remainingContracts must be 1 <p>* Step 6:</p> <p>Message: Get15118EVCertificateResponse</p> <ul style="list-style-type: none"> - status must be <i>Accepted</i> - exiResponse <Raw CertificateInstallationRes response for the EV, Base64 encoded.> - remainingContracts must be 0

Tool validations
Post scenario validations: N/a

TC_M_101_CSMS: Install CA certificate - OEMRootCertificate

Test case name	Install CA certificate - OEMRootCertificate
Test case Id	TC_M_101_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 OEMRootCertificate.
Prerequisite(s)	- The CSMS supports ISO15118-20

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

Main (Test scenario)	
Charging Station	CSMS
1. Execute Reusable State <i>CertificateInstalled</i> for certificateType <i>OEMRootCertificate</i>	

Tool validations
N/a
Post scenario validations: N/a

N Diagnostics

TC_N_01_CSMS: Get Monitoring Report - with monitoringCriteria

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. Test System responds with: GetMonitoringReportResponse with: Status <i>EmptyResultSet</i>	1. CSMS sends GetMonitoringReportRequest
Manually instruct CSMS to get a report of monitors for: - all ThresholdMonitoring	
4. Test System responds with: GetMonitoringReportResponse with: Status <i>Accepted</i>	3. CSMS sends GetMonitoringReportRequest
5. Test System 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 is omitted.
* Step 3: Message: GetMonitoringReportRequest - monitoringCriteria = ThresholdMonitoring - componentVariable is omitted.
Post scenario validations: Check that CSMS shows the <i>Threshold</i> monitors.

TC_N_02_CSMS: Get Monitoring Report - with component/variable

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. Test System responds with: GetMonitoringReportResponse with: Status <i>EmptyResultSet</i>	1. CSMS sends GetMonitoringReportRequest
Manually instruct CSMS to get a report of monitors for: - the variable AvailabilityState of EVSE #1.	
4. Test System responds with: GetMonitoringReportResponse with: Status <i>Accepted</i>	3. CSMS sends GetMonitoringReportRequest
5. Test System responds with: NotifyMonitoringReportRequest	6. CSMS sends NotifyMonitoringReportResponse
Step 5 and 6 are repeated as often as needed to report all configuration variables.	

Tool validations
<p>* Step 1:</p> <p>Message: GetMonitoringReportRequest</p> <ul style="list-style-type: none"> - componentVariable[0].component.name = "ChargingStation" - componentVariable[0].component.instance is omitted. - componentVariable[0].variable.name = "Power" - componentVariable[0].variable.instance is omitted. - monitoringCriteria is omitted.

Tool validations

* Step 3:

Message: **GetMonitoringReportRequest**

- **componentVariable[1].component.name** = "EVSE"
- **componentVariable[1].component.instance** is omitted.
- **componentVariable[1].component.evse.id** = 1
- **componentVariable[1].variable.name** = "AvailabilityState"
- **componentVariable[1].variable.instance** is omitted.
- **monitoringCriteria** is omitted.

Post scenario validations:

Check that CSMS shows the monitor for AvailabilityState for EVSE #1.

TC_N_03_CSMS: Get Monitoring Report - with component criteria and component/variable

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
<i>Manually instruct CSMS to get a report of monitors for:</i> - all <i>DeltaMonitoring</i> - and the variable <i>AvailabilityState</i> for EVSE #1.	
2. Test System responds with: GetMonitoringReportResponse with: Status <i>EmptyResultSet</i>	1. CSMS sends GetMonitoringReportRequest
<i>Manually instruct CSMS to get a report of monitors for:</i> - all <i>ThresholdMonitoring</i> - and the variable <i>Power</i> of ChargingStation.	
4. Test System responds with: GetMonitoringReportResponse with: Status <i>Accepted</i>	3. CSMS sends GetMonitoringReportRequest
5. Test System responds with: NotifyMonitoringReportRequest	6. CSMS sends NotifyMonitoringReportResponse
<i>Step 5 and 6 are repeated as often as needed to report all configuration variables.</i>	

Tool validations

Tool validations
<p>* Step 1:</p> <p>Message: GetMonitoringReportRequest</p> <ul style="list-style-type: none">- monitoringCriteria = <i>DeltaMonitoring</i>- componentVariable[0].component.name = "EVSE"- componentVariable[0].component.evse.id = <configured evseld>- componentVariable[0].variable.name = "AvailabilityState" <p>* Step 3:</p> <p>Message: GetMonitoringReportRequest</p> <ul style="list-style-type: none">- monitoringCriteria = <i>ThresholdMonitoring</i>- componentVariable[0].component.name = "ChargingStation"- componentVariable[0].variable.name = "Power"
<p>Post scenario validations:</p> <p>Check that CSMS shows the <i>Threshold</i> monitors for Power for ChargingStation.</p>

TC_N_60_CSMS: Get Monitoring Report - with component criteria and list of components/variables

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 <i>ThresholdMonitoring</i> - and the variable <i>Power</i> of both <i>ChargingStation</i> and <i>EVSE #1</i> .	
2. Test System responds with: GetMonitoringReportResponse with: Status <i>EmptyResultSet</i>	1. CSMS sends GetMonitoringReportRequest
Manually instruct CSMS to get a report of monitors for: - all <i>DeltaMonitoring</i> - and the variable <i>AvailabilityState</i> of both <i>ChargingStation</i> and <i>EVSE #1</i> .	
4. Test System responds with: GetMonitoringReportResponse with: Status <i>Accepted</i>	3. CSMS sends GetMonitoringReportRequest
5. Test System 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** 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.

TC_N_05_CSMS: Set Monitoring Base - success

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 <i>All</i> , <i>FactoryDefault</i> and <i>HardWiredOnly</i> .
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. Test System responds with: SetMonitoringBaseResponse	<i>Instruct CSMS to set a monitoring base of _All._</i> 1. CSMS sends SetMonitoringBaseRequest
4. Test System responds with: SetMonitoringBaseResponse	<i>Instruct CSMS to set a monitoring base of _FactoryDefault._</i> 3. Test System sends SetMonitoringBaseRequest
6. The Test System responds with: SetMonitoringBaseResponse	<i>Instruct CSMS to set a monitoring base of _HardWiredOnly._</i> 5. Test System 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

TC_N_08_CSMS: Set Variable Monitoring - One SetMonitoringData element

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)
<p>Configuration State: This test case activates monitoring on the following variable: - Component "EVSE", evse <Configured evseld>, variable "AvailabilityState", monitor type <i>Delta</i> It assumes, that no monitor is active on this variable prior to the test. <i>Note 1: this is a required variable for which a monitor can be expected to exist or it can be configured.</i> <i>Note 2: Any other component/variable combination that supports monitoring could also be used for this test case.</i></p> <p>Memory State: N/a</p> <p>Reusable State(s): N/a</p>

Main (Test scenario)	
Charging Station	CSMS
2. Test System responds with: Message: SetVariableMonitoringResponse with setMonitoringResult[0].status = <i>Accepted</i>	1. Request CSMS to install monitors on: - EVSE #<Configured evseld>, AvailabilityState, Delta, severity 8

Tool validations
<p>* Step 1:</p> <p>1. CSMS sends SetVariableMonitoringRequest with:</p> <ul style="list-style-type: none"> - setMonitoringData[0].value = 1, ← recommended value for <i>Delta</i> monitor - setMonitoringData[0].type = <i>Delta</i>, - setMonitoringData[0].severity = 8, - setMonitoringData[0].component.name = "EVSE" - setMonitoringData[0].component.evse.id = <Configured evseld> - setMonitoringData[0].variable.name = "AvailabilityState" <p>Post scenario validations: N/A</p>

TC_N_09_CSMS: Set Variable Monitoring - Multiple elements on different component and variable

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)
<p>Configuration State:</p> <p>This test case activates monitors on the following variables:</p> <ul style="list-style-type: none"> - Component "EVSE", evse <Configured evseld>, variable "AvailabilityState", monitor type <i>Delta</i> - Component "ChargingStation", variable "AvailabilityState", monitor type <i>Delta</i> <p>It assumes, that no monitor is active on these variables prior to the test.</p> <p><i>Note 1: these are required variables for which a monitor can be expected to exist or it can be configured.</i></p> <p><i>Note 2: Any other component/variable combination that supports monitoring could also be used for this test case.</i></p> <p>Memory State: N/a</p> <p>Reusable State(s): N/a</p>

Main (Test scenario)	
Charging Station	CSMS
2. Test System responds with: Message: SetVariableMonitoringResponse with setMonitoringResult[0].status = <i>Accepted</i>	1. Request CSMS to install monitors on: - EVSE #<Configured evseld>, AvailabilityState, <i>Delta</i> , severity 8 - ChargingStation, AvailabilityState, <i>Delta</i> , severity 8

Tool validations
<p>* Step 1:</p> <p>1. CSMS sends SetVariableMonitoringRequest with:</p> <ul style="list-style-type: none"> - setMonitoringData[0].value = 1, ← recommended value for <i>Delta</i> monitor - setMonitoringData[0].type = Delta, - setMonitoringData[0].severity = 8, - setMonitoringData[0].component.name = "EVSE" - setMonitoringData[0].component.evse.id = <Configured evseld> - setMonitoringData[0].variable.name = "AvailabilityState" <ul style="list-style-type: none"> - setMonitoringDate[1].value = 1, - setMonitoringDate[1].type = Delta, - setMonitoringDate[1].severity = 8, - setMonitoringDate[1].component.name = "ChargingStation" - setMonitoringDate[1].variable.name = "AvailabilityState" <p>Post scenario validations: N/A</p>

TC_N_16_CSMS: Set Monitoring Level - Success

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. Test System responds with: SetMonitoringLevelResponse with Status is <i>Accepted</i>	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

TC_N_17_CSMS: Set Monitoring Level - Out of range

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 Test System 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)	
Charging Station	CSMS
2. Test System responds with: SetMonitoringLevelResponse with Status is <i>Rejected</i>	1. <i>Instruct CSMS to set a monitoring level with severity = 4</i>

Tool validations
* Step 1: Message: SetMonitoringLevelRequest with: severity = 4
Post scenario validations: N/A

TC_N_18_CSMS: Clear Monitoring - Too many elements

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 <code>ItemsPerMessageClearVariableMonitoring</code> amount of monitors in one request.
Prerequisite(s)	CS has implemented device model monitoring and <code>MonitoringCtrlr.Enabled = true</code> .

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 Test System responds with: GetVariablesResponse	1. Instruct CSMS to send <i>GetVariablesRequest</i> with: Component.name <i>MonitoringCtrlr</i> Variable.name <i>ItemsPerMessage</i> Variable.instance <i>ClearVariableMonitoring</i> .
4. The Test System responds with: ClearVariableMonitoringResponse	3. Instruct CSMS to clear more monitors than allowed in <i>ItemsPerMessage</i> . ClearVariableMonitoringRequest with a list of ids <i>Note: these monitor ids do not have to exist.</i>

Tool validations

* Step 1:

Message: Two or more **ClearVariableMonitoringRequest**, so that the maximum number of `ItemsPerMessageClearVariableMonitoring` **ids** is never exceeded.

Test System will reply with a **ClearVariableMonitoringResponse** for each **ClearVariableMonitoringRequest**, but the content of the responses is irrelevant for the test.

Post scenario validations:

N/A

TC_N_21_CSMS: Alert Event - HardWiredMonitor

Test case name	Alert Event - HardWiredMonitor
Test case Id	TC_N_21_CSMS
Use case Id(s)	N07
Requirement(s)	N07.FR.03
System under test	CSMS
Description	Charging Station sends an NotifyEventRequest for a HardWiredMonitor.
Purpose	To test that the CSMS is able to handle a HardWiredMonitor.
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. Test System sends NotifyEventRequest message with eventNotificationType = <i>HardWiredMonitor</i>	2. CSMS returns NotifyEventResponse message.

Tool validations
* Step 2: Message: NotifyEventResponse with empty body.
Post scenario validations: N/A

TC_N_24_CSMS: Set Variable Monitoring - Periodic event

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. <i>Note: this is identical to TC_N_21_CSMS, only with a periodic event.</i>
Prerequisite(s)	N/a

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

Main (Test scenario)	
Charging Station	CSMS
Tester makes Test System send a NotifyEventRequest message.	
1. Test System sends NotifyEventRequest message.	2. CSMS returns NotifyEventResponse message.
<u>Note(s):</u> - Step 1 and 2 will be repeated n times	

Tool validations
* Step 2: Message: NotifyEventResponse with empty body.
Post scenario validations: N/A

TC_N_25_CSMS: Retrieve Log Information - Diagnostics Log - Success

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Test System responds with a GetLogResponse with status <i>Accepted</i>	1. The CSMS sends a GetLogRequest
3. The Test System sends a LogStatusNotificationRequest with - status <i>Uploading</i> - requestId <i>Same Id as the GetLogRequest</i>	4. The CSMS responds with a LogStatusNotificationResponse .
5. The Test System sends a LogStatusNotificationRequest with - status <i>Uploaded</i> - requestId <i>Same Id as the GetLogRequest</i>	6. The CSMS responds with a LogStatusNotificationResponse .

Tool validations
* Step 1: Message GetLogRequest - logType <i>DiagnosticsLog</i>
Post scenario validations: - N/a

TC_N_27_CSMS: Get Customer Information - Accepted + data

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Test System responds with a CustomerInformationResponse with status Accepted	1. The CSMS sends a CustomerInformationRequest
3. The Test System sends a NotifyCustomerInformationRequest	4. The CSMS responds with a NotifyCustomerInformationResponse .

Tool validations
<p>* Step 1:</p> <p>Message CustomerInformationRequest</p> <ul style="list-style-type: none"> - report true - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type>
<p>Post scenario validations:</p> <ul style="list-style-type: none"> - N/a

TC_N_28_CSMS: Get Customer Information - Accepted + no data

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Test System responds with a CustomerInformationResponse with status Accepted	1. The CSMS sends a CustomerInformationRequest
3. The Test System sends a NotifyCustomerInformationRequest	4. The CSMS responds with a NotifyCustomerInformationResponse .

Tool validations
<p>* Step 1:</p> <p>Message CustomerInformationRequest</p> <ul style="list-style-type: none"> - report true - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type>
<p>Post scenario validations:</p> <ul style="list-style-type: none"> - N/a

TC_N_29_CSMS: Get Customer Information - Not Accepted

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Test System responds with a CustomerInformationResponse with status <i>Rejected</i>	1. The CSMS sends a CustomerInformationRequest

Tool validations
<p>* Step 1:</p> <p>Message CustomerInformationRequest</p> <ul style="list-style-type: none"> - report true - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type>
<p>Post scenario validations:</p> <ul style="list-style-type: none"> - N/a

TC_N_30_CSMS: Clear Customer Information - Clear and report + data

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Test System responds with a CustomerInformationResponse with status <i>Accepted</i>	1. The CSMS sends a CustomerInformationRequest
3. The Test System sends a NotifyCustomerInformationRequest	4. The CSMS responds with a NotifyCustomerInformationResponse .

Tool validations
<p>* Step 1:</p> <p>Message CustomerInformationRequest</p> <ul style="list-style-type: none"> - report true - clear true - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type>
<p>Post scenario validations:</p> <ul style="list-style-type: none"> - N/a

TC_N_31_CSMS: Clear Customer Information - Clear and report + no data

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Test System responds with a CustomerInformationResponse with status Accepted	1. The CSMS sends a CustomerInformationRequest
3. The Test System sends a NotifyCustomerInformationRequest	4. The CSMS responds with a NotifyCustomerInformationResponse .

Tool validations
<p>* Step 1:</p> <p>Message CustomerInformationRequest</p> <ul style="list-style-type: none"> - report <i>true</i> - clear <i>true</i> - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type>
<p>Post scenario validations:</p> <ul style="list-style-type: none"> - N/a

TC_N_32_CSMS: Clear Customer Information - Clear and no report

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Test System responds with a CustomerInformationResponse with status <i>Accepted</i>	1. The CSMS sends a CustomerInformationRequest
3. The Test System sends a NotifyCustomerInformationRequest	4. The CSMS responds with a NotifyCustomerInformationResponse

Tool validations
<p>* Step 1:</p> <p>Message CustomerInformationRequest</p> <ul style="list-style-type: none"> - report <i>false</i> - clear <i>true</i> - idToken.idToken <Configured valid_idtoken_idtoken> - idToken.type <Configured valid_idtoken_type>
<p>Post scenario validations:</p> <ul style="list-style-type: none"> - N/a

TC_N_62_CSMS: Clear Customer Information - Clear and report - customerIdentifier

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 to the notifications as described in the OCPP specification.
Prerequisite(s)	The CSMS supports retrieving / deleting CustomerInformation - CustomerIdentifier

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Test System responds with a CustomerInformationResponse with status Accepted	1. The CSMS sends a CustomerInformationRequest
3. The Test System sends a NotifyCustomerInformationRequest	4. The CSMS responds with a NotifyCustomerInformationResponse

Tool validations
<p>* Step 1:</p> <p>Message CustomerInformationRequest</p> <ul style="list-style-type: none"> - report <i>true</i> - clear <i>true</i> - customerIdentifier <i>"OpenChargeAlliance"</i> - idToken is omitted - customerCertificate is omitted
Post scenario validations:
- N/a

TC_N_63_CSMS: Clear Customer Information - Clear and report - customerCertificate

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 to the notifications as described in the OCPP specification.
Prerequisite(s)	The CSMS supports retrieving / deleting CustomerInformation - CustomerCertificate

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Test System responds with a CustomerInformationResponse with status <i>Accepted</i>	1. The CSMS sends a CustomerInformationRequest with specific hash data <i><customer certificate hash data></i> .
3. The Test System sends a NotifyCustomerInformationRequest	4. The CSMS responds with a NotifyCustomerInformationResponse

Tool validations
* Step 1: Message CustomerInformationRequest - report true - clear true - customerCertificate contains <i><customer certificate hash data></i>
Post scenario validations: - N/a

TC_N_34_CSMS: Retrieve Log Information - Rejected

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Test System responds with a GetLogResponse with status <i>Rejected</i>	1. The CSMS sends a GetLogRequest

Tool validations
N/a
Post scenario validations: - N/a

TC_N_35_CSMS: Retrieve Log Information - Security Log - Success

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.

Reusable State:

N/a

Main (Test scenario)

Charging Station	CSMS
2. The Test System responds with a GetLogResponse with status Accepted	1. The CSMS sends a GetLogRequest
3. The Test System sends a LogStatusNotificationRequest with - status Uploading - requestId Same Id as the GetLogRequest	4. The CSMS responds with a LogStatusNotificationResponse .
5. The Test System sends a LogStatusNotificationRequest with - status Uploaded - requestId Same Id as the GetLogRequest	6. The Test System responds with a LogStatusNotificationResponse .

Tool validations

* Step 1:

Message **GetLogRequest**- **logType SecurityLog**

Post scenario validations:

- N/a

TC_N_36_CSMS: Retrieve Log Information - Second Request

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.

Reusable State:

N/a

Main (Test scenario)

Charging Station	CSMS
2. The Test System responds with a GetLogResponse with status Accepted	1. The CSMS sends a GetLogRequest
3. The Test System sends a LogStatusNotificationRequest with - status Uploading - requestId Same Id as the GetLogRequest from Step 1	4. The CSMS responds with a LogStatusNotificationResponse .
6. The Test System responds with a GetLogResponse with status AcceptedCanceled	5. The CSMS sends a GetLogRequest
7. The Test System sends a LogStatusNotificationRequest with - status AcceptedCanceled - requestId Same Id as the GetLogRequest from Step 1	8. The CSMS responds with a LogStatusNotificationResponse .
9. The Test System sends a LogStatusNotificationRequest with - status Uploading - requestId Same Id as the GetLogRequest from Step 5	10. The CSMS responds with a LogStatusNotificationResponse .
11. The Test System 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

TC_N_44_CSMS: Clear Monitoring - Rejected

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Test System responds with a ClearVariableMonitoringResponse with clearMonitoringResult[0].status <i>Rejected</i>	1. The CSMS sends a ClearVariableMonitoringRequest

Tool validations
N/a
Post scenario validations: - N/a

TC_N_46_CSMS: Clear Customer Information - Update Local Authorization List

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 customer 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 valid_idtoken_idtoken> is configured.
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
<u>Manual action:</u> Trigger CSMS to CustomerInformationRequest to both report and clear token <Configured valid_idtoken_idtoken> and <Configured valid_idtoken_type>	
2. The Test System responds with a CustomerInformationResponse with status Accepted	1. The CSMS sends a CustomerInformationRequest
3. The Test System sends a NotifyCustomerInformationRequest	4. The CSMS responds with a NotifyCustomerInformationResponse .
<u>Manual action:</u> If not triggered automatically, trigger CSMS to send SendLocalListRequest with version = <configured local list version> + 1 and updateType = Differential and localAuthorizationList = [{idToken = {<Configured_valid_idtoken>, <Configured valid_idtoken_type>}}]	
6 The Test System responds with a SendLocalListResponse with status Accepted	5. The CSMS sends a SendLocalListRequest

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 <configured local list version> + 1 - localAuthorizationListp[0].idToken contains <configured_valid_idtoken_idtoken> and <configured valid_idtoken_type> - localAuthorizationList[0].idTokenInfo <omitted>

Tool validations
Post scenario validations: - All messages have been received

TC_N_47_CSMS: Get Monitoring report - Report all

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Test System responds with a GetMonitoringReportResponse	1. The CSMS sends a GetMonitoringReportRequest
3. The Test System sends a NotifyMonitoringReportRequest	4. The CSMS responds with a NotifyMonitoringReportResponse .
<u>Note(s):</u> - 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

TC_N_48_CSMS: Alert Event - Variable monitoring on write only

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
1. The Test System sends a NotifyEventRequest with eventData.actualValue empty	2. The CSMS responds with a NotifyEventResponse .

Tool validations
N/a
Post scenario validations: - N/a

TC_N_49_CSMS: Alert Event - LowerThreshold/UpperThreshold cleared after reboot

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
1. The Test System sends a NotifyEventRequest with eventData.cleared <i>true</i>	2. The CSMS responds with a NotifyEventResponse .

Tool validations
N/a
Post scenario validations: - N/a

TC_N_50_CSMS: Alert Event - Periodic Triggered

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
1. The Test System sends a NotifyEventRequest with eventData.trigger <i>Periodic</i>	2. The CSMS responds with a NotifyEventResponse .

Tool validations
N/a
Post scenario validations: - N/a

TC_N_100_CSMS: Retrieve Log Information - DataCollectorLog - Success

Test case name	Retrieve Log Information - DataCollectorLog - Success
Test case Id	TC_N_100_CSMS
Use case Id(s)	N01
Requirement(s)	...
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 CSMS is able to successfully retrieve a log of logType <i>DataCollectorLog</i> as described at the OCPP specification.
Prerequisite(s)	<ul style="list-style-type: none"> - CSMS supports DataCollectorLog. - 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). - Charging station supports DataCollector (Component DataCollector available).

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action:</u> Request the CSMS to retrieve the DataCollectorLog	
	1. The CSMS sends a GetLogRequest
2. The Test System responds with a GetLogResponse	
3. The Test System sends a LogStatusNotificationRequest with status <i>Uploading</i> requestId <GetLogRequest.requestId of step 1>	4. The CSMS responds with a LogStatusNotificationResponse .
5. The Test System sends a LogStatusNotificationRequest with status <i>Uploaded</i> requestId <GetLogRequest.requestId of step 1>	6. The CSMS responds with a LogStatusNotificationResponse .

Tool validations
* Step 1: Message GetLogRequest - logType <i>DataCollectorLog</i>
Post scenario validations: N/a

TC_N_102_CSMS: Retrieve Log Information - Authentication - HTTP

Test case name	Retrieve Log Information - Authentication - HTTP
Test case Id	TC_N_102_CSMS
Use case Id(s)	N01
Requirement(s)	N01.FR.26, N01.FR.27, N01.FR.28
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 successfully request and process a log of logType <i>DiagnosticsLog</i> as described at the OCPP specification.
Prerequisite(s)	- CSMS supports HTTP file transfer protocol. - A diagnostics logging server has been set up for HTTP and accepts specific userinfo credentials .

Before (Preparations)
Configuration State: CSMS configured to prevent unauthorised uploading to HTTP log location.
Memory State: N/a
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
<u>Manual action:</u> Trigger CSMS to <i>GetLogRequest</i> with logType <i>DiagnosticsLog</i> and remoteLocation with userinfo for HTTP Basic Authentication	
2. Test System responds with a GetLogResponse with status <i>Accepted</i>	1. CSMS sends a GetLogRequest
3. The Test System uploads log file to <GetLogRequest.log.remoteLocation of step 1> using HTTP POST	4. The CSMS responds using HTTP
5. The Test System uploads file to <GetLogRequest.log.remoteLocation of step 1> using HTTP POST	6. The CSMS responds using HTTP
7. Test System sends a LogStatusNotificationRequest with status <i>Uploading</i> requestId <GetLogRequest.requestId of step 1>	8. CSMS responds with a LogStatusNotificationResponse
9. Test System sends a LogStatusNotificationRequest with status <i>Uploaded</i> requestId <GetLogRequest.requestId of step 1>	10. CSMS responds with a LogStatusNotificationResponse

Tool validations
<p>* Step 1:</p> <p>Message GetLogRequest</p> <ul style="list-style-type: none">- logType must be <i>DiagnosticsLog</i>- log.remoteLocation must<ul style="list-style-type: none">- contain a valid URL- specify scheme <i>HTTP</i>- contain userinfo for HTTP Basic Authentication- not contain a URL fragment component- not contain a URL query components that ends in a slash character ("/") <p>* Step 4:</p> <ul style="list-style-type: none">- HTTP status code must be <i>401 Unauthorized</i> <p>* Step 6:</p> <ul style="list-style-type: none">- HTTP status code must be in 2xx-range
<p>Post scenario validations:</p> <p>N/a</p>

TC_N_103_CSMS: Retrieve Log Information - Authentication - HTTPS

Test case name	Retrieve Log Information - Authentication - HTTPS
Test case Id	TC_N_102_CSMS
Use case Id(s)	N01
Requirement(s)	N01.FR.26, N01.FR.27, N01.FR.28
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 successfully request and process a log of logType <i>DiagnosticsLog</i> as described at the OCPP specification.
Prerequisite(s)	- CSMS supports HTTPS file transfer protocol. - A diagnostics logging server has been set up for HTTPS and accepts specific userinfo credentials .

Before (Preparations)
Configuration State: CSMS configured to prevent unauthorised uploading to HTTPS log location.
Memory State: N/a
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
<u>Manual action:</u> Trigger CSMS to <i>GetLogRequest</i> with logType <i>DiagnosticsLog</i> and remoteLocation with userinfo for HTTP Basic Authentication	
2. Test System responds with a GetLogResponse with status <i>Accepted</i>	1. CSMS sends a GetLogRequest
3. The Test System uploads log file to <i><GetLogRequest.log.remoteLocation of step 1></i> using HTTP POST	4. The CSMS responds using HTTP
5. The Test System uploads file to <i><GetLogRequest.log.remoteLocation of step 1></i> using HTTP POST	6. The CSMS responds using HTTP
7. Test System sends a LogStatusNotificationRequest with status <i>Uploading</i> requestId <i><GetLogRequest.requestId of step 1></i>	8. CSMS responds with a LogStatusNotificationResponse
9. Test System sends a LogStatusNotificationRequest with status <i>Uploaded</i> requestId <i><GetLogRequest.requestId of step 1></i>	10. CSMS responds with a LogStatusNotificationResponse

Tool validations
<p>* Step 1:</p> <p>Message GetLogRequest</p> <ul style="list-style-type: none">- logType must be <i>DiagnosticsLog</i>- log.remoteLocation must<ul style="list-style-type: none">- contain a valid URL- specify scheme <i>HTTPS</i>- contain userinfo for HTTP Basic Authentication- not contain a URL fragment component- not contain a URL query components that ends in a slash character ("/") <p>* Step 4:</p> <ul style="list-style-type: none">- HTTP status code must be <i>401 Unauthorized</i> <p>* Step 6:</p> <ul style="list-style-type: none">- HTTP status code must be in 2xx-range
<p>Post scenario validations:</p> <p>N/a</p>

TC_N_104_CSMS: Get Monitoring report - TargetDeltaMonitoring

Test case name	Get Monitoring report - TargetDeltaMonitoring
Test case Id	TC_N_104_CSMS
Use case Id(s)	N02
Requirement(s)	...
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: N/a
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
<u>Manual action:</u> <i>Trigger CSMS to get a report of monitors for DeltaMonitoring</i>	
2. Test System responds with a GetMonitoringReportResponse with Status <i>Accepted</i>	1. CSMS sends GetMonitoringReportRequest
3. Test System responds with a NotifyMonitoringReportRequest with requestId <i><GetMonitoringReportRequest.requestId from step 1></i> tbc <i>false</i> seqNo <i>0</i> monitor[0].component.name <i>EVSE</i> monitor[0].component.evse.id <i>1</i> monitor[0].variable.name <i>Power</i> monitor[0].variableMonitoring[0].type <i>TargetDeltaRelative</i> monitor[0].variableMonitoring[0].transaction = <i>false</i> monitor[0].variableMonitoring[0].value = <i>5</i> monitor[0].variableMonitoring[0].eventNotificationType = <i><not omitted></i> monitor[0].variableMonitoring[1].type <i>TargetDelta</i> monitor[0].variableMonitoring[1].transaction = <i>false</i> monitor[0].variableMonitoring[1].value = <i>5</i> monitor[0].variableMonitoring[1].eventNotificationType = <i><not omitted></i>	4. CSMS sends NotifyMonitoringReportResponse

Tool validations
* Step 1: Message: GetMonitoringReportRequest - monitoringCriteria = <i>DeltaMonitoring</i>
Post scenario validations: N/a

TC_N_105_CSMS: Set Variable Monitoring - Set Frequent Periodic Variable Monitoring - Periodic

Test case name	Set Variable Monitoring - Set Frequent Periodic Variable Monitoring - Periodic
Test case Id	TC_N_105_CSMS
Use case Id(s)	N11, N14, N15
Requirement(s)	N11.FR.06, N11.FR.08, N14.FR.01, N15.FR.02
System under test	CSMS
Description	To give the CSMS the ability to request efficient frequent periodic monitoring of variables.
Purpose	To verify if the CSMS can accept and receive periodic event streams.
Prerequisite(s)	- The CSMS supports periodic event streams.

Before (Preparations)

Configuration State:
CSMS must be configured to accept opening the periodic event stream.

Memory State:
N/a

Reusable State:
N/a

Main (Test scenario)

Charging Station	CSMS
1. Test System sends a OpenPeriodicEventStreamRequest with constantStreamData.id 2 constantStreamData.variableMonitoringId 3 constantStreamData.params.interval 10 constantStreamData.params.values 30	2. CSMS responds with a OpenPeriodicEventStreamResponse
Manual Action: Request the CSMS to reconfigure the periodic event stream to an interval of 15s and values 45	
4. Test System responds with a AdjustPeriodicEventStreamResponse with status <i>Accepted</i>	3. CSMS sends a AdjustPeriodicEventStreamRequest
5. Charging Station sends a NotifyPeriodicEventStream using RFC framework SEND	
6. Test System sends a ClosePeriodicEventStreamRequest with constantStreamData.id 2	7. CSMS responds with a OpenPeriodicEventStreamResponse
Note: The Test System waits 10 seconds, before ending the testcase.	

Tool validations

* Step 2:
Message **OpenPeriodicEventStreamResponse**
- **status** must be *Accepted*

* Step 3:
Message **AdjustPeriodicEventStreamRequest**
id must be 2
params.interval must be 15
params.values must be 45

Post scenario validations:
- CSMS does not respond with any message when receiving message **NotifyPeriodicEventStream** of step 5.

TC_N_107_CSMS: Set Variable Monitoring - Get Periodic Event Streams - Goodflow

Test case name	Set Variable Monitoring - Get Periodic Event Streams - Goodflow
Test case Id	TC_N_107_CSMS
Use case Id(s)	N12
Requirement(s)	N12.FR.01
System under test	CSMS
Description	To get a list of existing event streams
Purpose	To verify if the CSMS can request retrieving a list of periodic event streams.
Prerequisite(s)	- The CSMS supports periodic event streams.

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action:</u> Request the CSMS to retrieve a list of existing event streams from the Charging Station	
<p>2. Test System responds with a GetPeriodicEventStreamResponse with</p> <p>constantStreamData[0].id 2</p> <p>constantStreamData[0].variableMonitoringId 3</p> <p>constantStreamData[0].params.interval 10</p> <p>constantStreamData[0].params.values 30</p> <p>constantStreamData[1].id 3</p> <p>constantStreamData[1].variableMonitoringId 4</p> <p>constantStreamData[1].params.interval 10</p> <p>constantStreamData[1].params.values 30</p>	<p>1. CSMS sends a GetPeriodicEventStreamRequest</p>

Tool validations
N/a
Post scenario validations: N/a

0 Display Message

TC_O_01_CSMS: Set Display Message - Success

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 send a 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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action:</u> Request the CSMS to send a SetDisplayMessageRequest.	
2. The Test System 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

TC_O_02_CSMS: Get all Display Messages - Success

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 the 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.
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Test System responds with a GetDisplayMessagesResponse with status Accepted	1. The CSMS sends a GetDisplayMessagesRequest
3. The Test System 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

TC_O_03_CSMS: Get all Display Messages - No DisplayMessages configured

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 the 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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Test System 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

TC_O_04_CSMS: Clear Display Message - Success

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.
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
<u>Note</u> : As a help method, a <i>GetDisplayMessagesRequest</i> is requested first for CSMS's that implemented their <i>ClearDisplayMessage</i> as a combined feature.	
2. The Test System responds with a ClearDisplayMessageResponse with status <i>Accepted</i>	1. The CSMS sends a ClearDisplayMessageRequest

Tool validations
* Step 1: Message ClearDisplayMessageRequest - id <Generated Id from set display message>
Post scenario validations: - N/a

TC_O_05_CSMS: Clear Display Message - Unknown Key

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Test System responds with a ClearDisplayMessageResponse with status <i>Unknown</i>	1. The CSMS sends a ClearDisplayMessageRequest

Tool validations
N/a
Post scenario validations: - N/a

TC_O_06_CSMS: Set Display Message - Specific transaction - Success

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
Reusable State: State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action</u> : Request the CSMS to send a display message for a specific transaction.	
2. The Test System responds with a SetDisplayMessageResponse with status <i>Accepted</i>	1. The CSMS sends a SetDisplayMessageRequest
3. Execute Reusable State <i>EVDisconnected</i>	

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

TC_O_07_CSMS: Get a Specific Display Message - Id

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.
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Test System responds with a GetDisplayMessagesResponse with status Accepted	1. The CSMS sends a GetDisplayMessagesRequest
3. The Test System sends a NotifyDisplayMessagesRequest	4. The CSMS responds with a NotifyDisplayMessagesResponse

Tool validations
<p>* Step 1:</p> <p>Message GetDisplayMessagesRequest</p> <ul style="list-style-type: none"> - id <Configured_Id> - priority <Omitted> - state <Omitted> - requestId <Generated Id>
<p>Post scenario validations:</p> <ul style="list-style-type: none"> - N/a

TC_O_08_CSMS: Get a Specific Display Message - Priority

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Test System responds with a GetDisplayMessagesResponse with status <i>Accepted</i>	1. The CSMS sends a GetDisplayMessagesRequest
3. The Test System 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

TC_O_09_CSMS: Get a Specific Display Message - State

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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Test System responds with a GetDisplayMessagesResponse with status <i>Accepted</i>	1. The CSMS sends a GetDisplayMessagesRequest
3. The Test System 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

TC_O_10_CSMS: Set Display Message - Specific transaction - UnknownTransaction

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
Reusable State: State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action</u> : Request the CSMS to send a display message for a specific transaction.	
2. The Test System responds with a SetDisplayMessageResponse with status <i>UnknownTransaction</i>	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

TC_O_11_CSMS: Get a Specific Display Message - Unknown parameters

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.
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Test System 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

TC_O_12_CSMS: Set Display Message - Replace DisplayMessage

Test case name	Set Display Message - Replace DisplayMessage
Test case Id	TC_O_12_CSMS
Use case Id(s)	006
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.
Reusable 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 Test System responds with a SetDisplayMessageResponse with status <i>Accepted</i>	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

TC_O_13_CSMS: Set Display Message - Display message at StartTime

Test case name	Set Display Message - Display message at StartTime
Test case Id	TC_O_13_CSMS
Use case Id(s)	001
Requirement(s)	001_FR_05
System under test	CSMS
Description	This test case describes how the CSMS can be requested to send a 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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action</u> : Request the CSMS to send a SetDisplayMessageRequest with a startTime.	
2. The Test System 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

TC_O_14_CSMS: Set Display Message - Remove message after EndTime

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 send a 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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action:</u> Request the CSMS to send a SetDisplayMessageRequest with a endTime.	
2. The Test System responds with a SetDisplayMessageResponse with status Accepted	1. The CSMS sends a SetDisplayMessageRequest

Tool validations
* Step 1: Message SetDisplayMessageRequest - message.id <Generated Id> - message.endTime <Configured endTime>
Post scenario validations: - N/a

TC_O_17_CSMS: Set Display Message - NotSupportedPriority

Test case name	Set Display Message - NotSupportedPriority
Test case Id	TC_O_17_CSMS
Use case Id(s)	001
Requirement(s)	N/a
System under test	CSMS
Description	This test case describes how the CSMS can be requested to send a 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

Reusable State:

N/a

Main (Test scenario)

Charging Station	CSMS
2. The Test System 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

TC_O_18_CSMS: Set Display Message - NotSupportedState

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 send a 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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Test System responds with a SetDisplayMessageResponse with status <i>NotSupportedState</i>	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

TC_O_19_CSMS: Set Display Message - NotSupportedMessageFormat

Test case name	Set Display Message - NotSupportedMessageFormat
Test case Id	TC_O_19_CSMS
Use case Id(s)	001
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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Test System responds with a SetDisplayMessageResponse with status <i>NotSupportedMessageFormat</i>	1. The CSMS sends a SetDisplayMessageRequest

Tool validations
* Step 1: Message SetDisplayMessageRequest - message.id <Generated Id>
Post scenario validations: - N/a

TC_O_25_CSMS: Set Display Message - Send Specific state

Test case name	Set Display Message - Send Specific state
Test case Id	TC_O_25_CSMS
Use case Id(s)	001
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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
2. The Test System responds with a SetDisplayMessageResponse with status <i>Accepted</i>	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

TC_O_26_CSMS: Set Display Message - Rejected

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 send a 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
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action:</u> Request the CSMS to send a <code>SetDisplayMessageRequest</code> with a Normal Cycle priority.	
2. The Test System responds with a <code>SetDisplayMessageResponse</code> with status <i>Rejected</i>	1. The CSMS sends a <code>SetDisplayMessageRequest</code>

Tool validations
* Step 1: Message SetDisplayMessageRequest - message.id <Generated Id> - message.priority <Configured Priority>
Post scenario validations: - N/a

TC_O_27_CSMS: Set Display Message - Specific transaction - Display message at StartTime

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 send a 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
Reusable State: State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
2. The Test System responds with a SetDisplayMessageResponse with status <i>Accepted</i>	1. The CSMS sends a SetDisplayMessageRequest

Tool validations
<p>* Step 1:</p> <p>Message SetDisplayMessageRequest</p> <ul style="list-style-type: none"> - message.state is <omitted> - message.startDateTime is <Configured startDateTime> - message.endDateTime is <omitted> - message.transactionId is <Generated transactionId from Before>
<p>Post scenario validations:</p> <ul style="list-style-type: none"> - N/a

TC_O_28_CSMS: Set Display Message - Specific transaction - Remove message after EndTime

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 send a 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 an 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
Reusable State: State is <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
2. The Test System responds with a SetDisplayMessageResponse with <i>status Accepted</i>	1. The CSMS sends a SetDisplayMessageRequest

Tool validations
<p>* Step 1:</p> <p>Message SetDisplayMessageRequest</p> <ul style="list-style-type: none"> - message.state is <omitted> - message.startDateTime is <omitted> - message.endDateTime is <Configured endDateTime> - message.transactionId is <Generated transactionId from Before>
<p>Post scenario validations:</p> <ul style="list-style-type: none"> - N/a

TC_O_100_CSMS: Set Display Message - unsupported language

Test case name	Set Display Message - unsupported language
Test case Id	TC_O_100_CSMS
Use case Id(s)	001
Requirement(s)	N/a
System under test	CSMS
Description	This test case describes how the CSMS can be requested to send a 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 handle a rejection request for DisplayMessage mechanism as result of an unsupported language status.
Prerequisite(s)	CSMS support multiple languages.

Before (Preparations)

Configuration State:

N/a

Memory State:

N/a

Reusable State:

N/a

Main (Test scenario)

Charging Station	CSMS
<u>Manual Action:</u> Request the CSMS to send a SetDisplayMessageRequest with a certain language. Test System will respond that the language is not supported.	
2. The Test System responds with a SetDisplayMessageResponse with status LanguageNotSupported	1. The CSMS sends a SetDisplayMessageRequest
<u>Manual Action:</u> Verify that CSMS UI reports that message was not accepted due to unknown language	

Tool validations

* Step 1:

Message SetDisplayMessageRequest

- message.id <Generated Id>

- message.priority <Configured Priority>

- message.message.language <Not omitted>

Post scenario validations:

- N/a

TC_O_101_CSMS: Set Display Message - Language preference of the EV Driver

Test case name	Set DisplayMessage - Language preference of the EV Driver
Test case Id	TC_O_101_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 send a 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 DisplayMessage with the default and a different language.
Prerequisite(s)	CSMS support multiple languages.

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State: N/a

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action:</u> Request the CSMS to send a <code>SetDisplayMessageRequest</code> with an additional language (other than the default language), that is supported on the charging station.	
2. The Test System responds with a <code>SetDisplayMessageResponse</code> with status <i>Accepted</i>	1. The CSMS sends a <code>SetDisplayMessageRequest</code>

Tool validations
<p>* Step 1:</p> <p>Message SetDisplayMessageRequest</p> <ul style="list-style-type: none"> - message.id <Generated Id> - message.priority <Configured Priority> - message.message.language <Not omitted default language> - message.message.content <Not omitted default language content> - message.messageExtra[0].language <Not omitted other language> - message.messageExtra[0].content <Not omitted other language content>
<p>Post scenario validations:</p> <ul style="list-style-type: none"> - N/a

P Data Transfer

TC_P_02_CSMS: Data Transfer to the CSMS - Rejected / Unknown VendorId / Unknown MessageId

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 Test System 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 <i>UnknownVendorId</i> OR <i>UnknownMessageId</i> OR <i>Rejected</i> (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

TC_P_03_CSMS: CustomData - Receive custom data

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 Test System sends a StatusNotificationRequest with customData <customData>	2. The CSMS responds with a StatusNotificationResponse
3. The Test System 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

Q Bidirectional Power Transfer

TC_Q_102_CSMS: V2X Authorisation - ISO15118-20 - Charging only (V2X control) before starting V2X - Allowed Energy Transfer modes omitted

Test case name	V2X Authorisation - ISO15118-20 - Charging only (V2X control) before starting V2X - Allowed Energy Transfer modes omitted
Test case Id	TC_Q_102_CSMS
Use case Id(s)	Q01
Requirement(s)	Q01.FR.08
System under test	CSMS
Description	Authorization of an EV by the CSMS to start a V2X power transfer loop.
Purpose	To check if the CSMS is able to provide an empty allowedEnergyTransfer list if it's not able to determine a list for allowedEnergyTransfer.
Prerequisite(s)	CSMS supports ISO15118-20 CSMS can be configured of influenced so it is not able to determine a list of allowedEnergyTransfer.

Before (Preparations)
Configuration State: - Configure the CSMS so that it is unable to determine a list of allowedEnergyTransfer.
Memory State: N/a
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
<i>Prepare CSMS with needed meta data from device model</i>	
<u>Manual action:</u> trigger CSMS to get the base report	
2. The Test System responds with a GetBaseReportResponse with status Accepted	1. The CSMS sends a GetBaseReportRequest
3. The Test System sends a NotifyReportRequest with requestId <GetBaseReportRequest.requestId of step 1> reportData[0].component.name V2XChargingCtrlr reportData[0].component.evse.id 1 reportData[0].variable.name V2XEnabled reportData[0].variableAttribute.value true	4. The CSMS responds with a NotifyReportResponse
<i>Start transaction</i>	
5. The Test System sends an AuthorizeRequest	6. The CSMS responds with an AuthorizeResponse

Tool validations
* Step 2: Message: GetBaseReportRequest - reportBase must be <i>FullInventory</i> * Step 6: Message: AuthorizeResponse - idTokenInfo.status must be <i>Accepted</i> - allowedEnergyTransfer must be <omitted>

Tool validations
Post scenario validations: N/a

TC_Q_103_CSMS: V2X Authorisation - ISO15118-20 - Charging needs rejected

Test case name	V2X Authorisation - ISO15118-20 - Charging needs rejected
Test case Id	TC_Q_103_CSMS
Use case Id(s)	Q01
Requirement(s)	Q01.FR.04
System under test	CSMS
Description	Authorization of an EV by the CSMS to start a V2X power transfer loop.
Purpose	To check if the CSMS rejects a NotifyEVChargingNeedsRequest when the EV requests a EnergyTransferModes that is not allowed.
Prerequisite(s)	CSMS supports ISO15118-20

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is ISO15118-20 transaction just started exposing: - <allowedEnergyTransferModes> - <transactionId>

Main (Test scenario)	
Charging Station	CSMS
Agree on energy transfer mode	
1. Test System sends a NotifyEVChargingNeedsRequest with evseld <Configured evseld> chargingNeeds.requestedEnergyTransfer one different than mentioned in <allowedEnergyTransferModes> or if omitted AC_BPT chargingNeeds.availableEnergyTransfer [AC_single_phase, AC_two_phase, AC_three_phase, AC_BPT, AC_BPT_DER, AC_DER] chargingNeeds.controlMode ScheduledControl chargingNeeds.v2xChargingParameters.maxChargePower 1234W chargingNeeds.v2xChargingParameters.maxDischargePower 0W	2. CSMS SHALL respond with a NotifyEVChargingNeedsResponse

Tool validations
* Step 2: Message: NotifyEVChargingNeedsResponse - status must be <i>Rejected</i>
Post scenario validations: N/a

TC_Q_107_CSMS: V2X Authorisation - ISO15118-20 - Charging only (V2X control) before starting V2X

Test case name	V2X Authorisation - ISO15118-20 - Charging only (V2X control) before starting V2X
Test case Id	TC_Q_107_CSMS
Use case Id(s)	Q02
Requirement(s)	Q02.FR.01, Q02.FR.02, Q02.FR.05
System under test	CSMS
Description	To start a transaction with operationMode ChargingOnly, such that it can become a bidirectional session at a later time.
Purpose	To check if the CSMS supports starting in a non-bidirectional way.
Prerequisite(s)	CSMS supports ISO15118-20

Before (Preparations)
Configuration State: - CSMS is configured to start transactions first in a non-V2X mode before upgrading to V2X
Memory State: N/a
Reusable State(s): State is ISO15118-20 transaction just started exposing: - <allowedEnergyTransferModes> - <transactionId>

Main (Test scenario)	
Charging Station	CSMS
Agree on energy transfer mode	
1. The Test System sends a NotifyEVChargingNeedsRequest with evseld <Configured evseld> chargingNeeds.requestedEnergyTransfer one of non-bidirectional mode in <allowedEnergyTransferModes> received in AuthorizeResponse or if omitted AC_three_phase chargingNeeds.availableEnergyTransfer [AC_single_phase, AC_two_phase, AC_three_phase, AC_BPT, AC_BPT_DER, AC_DER] chargingNeeds.controlMode ScheduledControl chargingNeeds.v2xChargingParameters.maxChargePower 1234W chargingNeeds.v2xChargingParameters.maxDischargePower 1234W	2. The CSMS responds with a NotifyEVChargingNeedsResponse
Setting charging profile	
4. The Test System responds with a SetChargingProfileResponse with status Accepted	3. The CSMS sends a SetChargingProfileRequest
<u>Manual Action:</u> Trigger CSMS to allow V2X after initial start	
6. The Test System sends a NotifyAllowedEnergyTransferResponse with	5. The CSMS sends a NotifyAllowedEnergyTransferRequest

Tool validations
<div><div>* Step 2:</div><div>Message: NotifyEVChargingNeedsResponse</div><div><div>- status must be <i>Accepted</i> or <i>Processing</i></div></div></div> <div><div>* Step 3:</div><div>Message: SetChargingProfileRequest</div><div><div>- evseld must be <i><Configured evseld></i></div><div>- chargingProfile.transactionId must be <i><transactionId></i></div><div>- chargingProfile.chargingSchedule[*].chargingSchedulePeriod[*].operationMode must be <i>ChargingOnly</i> <i><omitted></i></div><div>- chargingProfile.chargingSchedule[*].chargingSchedulePeriod[*].dischargeLimit must be <i><omitted></i> <i><empty></i></div><div>- chargingProfile.chargingSchedule[*].chargingSchedulePeriod[*].dischargeLimit_L2 must be <i><omitted></i> <i><empty></i></div><div>- chargingProfile.chargingSchedule[*].chargingSchedulePeriod[*].dischargeLimit_L3 must be <i><omitted></i> <i><empty></i></div><div>- chargingProfile.chargingSchedule[*].chargingSchedulePeriod[*].setpoint must be <i><omitted></i> <i><empty></i></div><div>- chargingProfile.chargingSchedule[*].chargingSchedulePeriod[*].setpoint_L2 must be <i><omitted></i> <i><empty></i></div><div>- chargingProfile.chargingSchedule[*].chargingSchedulePeriod[*].setpoint_L3 must be <i><omitted></i> <i><empty></i></div><div>- chargingProfile.chargingSchedule[*].chargingSchedulePeriod[*].setPointReactive must be <i><omitted></i> <i><empty></i></div><div>- chargingProfile.chargingSchedule[*].chargingSchedulePeriod[*].setPointReactive_L2 must be <i><omitted></i> <i><empty></i></div><div>- chargingProfile.chargingSchedule[*].chargingSchedulePeriod[*].setPointReactive_L3 must be <i><omitted></i> <i><empty></i></div></div></div> <div><div>* Step 5:</div><div>Message: NotifyAllowedEnergyTransferRequest</div><div><div>- transactionId must be <i><transactionId></i></div><div>- allowedEnergyTransfer must contain one of <i>[DC_BPT, AC_BPT, AC_BPT_DER]</i></div></div></div>
<div><div>Post scenario validations:</div><div>N/a</div></div>

TC_Q_108_CSMS: V2X Authorisation - ISO15118-20 - Central V2X control with charging schedule

Test case name	V2X Authorisation - ISO15118-20 - Central V2X control with charging schedule
Test case Id	TC_Q_108_CSMS
Use case Id(s)	Q03
Requirement(s)	Q03.FR.01, Q03.FR.02
System under test	CSMS
Description	To allow the CSMS to control the charge and discharge behaviour of an EV with power profiles.
Purpose	To verify the CSMS is able to set a charging profile with a setpoint in the ISO15118-20 transaction.
Prerequisite(s)	CSMS supports ISO15118-20

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is ISO15118-20 transaction started in V2X AC_BPT energy transfer mode exposing: - <allowedEnergyTransferModes> - <transactionId>

Main (Test scenario)	
Charging Station	CSMS
2. Test System responds with a SetChargingProfileResponse with status <i>Accepted</i>	1. CSMS sends a SetChargingProfileRequest

Tool validations
<p>* Step 1:</p> <p>Message: SetChargingProfileRequest</p> <ul style="list-style-type: none"> - evseld must be <Configured evseld> - chargingProfile.transactionId must be <transactionId> - chargingProfile.chargingProfilePurpose must be <i>TxProfile</i> - chargingProfile.chargingSchedule.chargingRateUnit must be <i>W</i> - chargingProfile.chargingSchedule.chargingSchedulePeriod[0].startPeriod must be <i>0</i> - chargingProfile.chargingSchedule.chargingSchedulePeriod[0].operationMode must be <i>CentralSetpoint</i> - chargingProfile.chargingSchedule.chargingSchedulePeriod[0].setpoint must be <i>10.000</i> - chargingProfile.chargingSchedule.chargingSchedulePeriod[0].numberPhases may be present - chargingProfile.chargingSchedule.chargingSchedulePeriod[0].phaseToUse may be present - all other fields in chargingProfile.chargingSchedule.chargingSchedulePeriod[0] must be absent
Post scenario validations: N/a

TC_Q_109_CSMS: Central V2X control with dynamic CSMS setpoint - push

Test case name	Central V2X control with dynamic CSMS setpoint - push
Test case Id	TC_Q_109_CSMS
Use case Id(s)	Q04, K28
Requirement(s)	K28.FR.01, K28.FR.02
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 specify the power or current drawn by EV.
Purpose	To verify if the CSMS is able to support DynamicControl charging profiles and push changes to the CS.
Prerequisite(s)	- CSMS supports Bidirectional Power Transfer

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 send a TxDefaultProfile dynamic charging profile to the CS which will be periodically updated every <Configured dynamic profile update interval> seconds by CSMS with limit $6 * \text{<limit multiplier>}$ Note: Check [csms_csKSmartChargingChargingProfileLimits] for <limit multiplier>	
2. The Test System responds with a SetChargingProfileResponse with status Accepted	1. The CSMS sends a SetChargingProfileRequest
4. The Test System responds with a UpdateDynamicScheduleResponse with status Accepted	3. The CSMS sends a UpdateDynamicScheduleRequest
6. The Test System responds with a UpdateDynamicScheduleResponse with status Accepted	5. The CSMS sends a UpdateDynamicScheduleRequest
<u>Manual Action:</u> Trigger the CSMS to clear the chargingProfile from step 1.	
8. The Test System responds with a ClearChargingProfileResponse with status Accepted	7. The CSMS sends a ClearChargingProfileRequest

Tool validations
<p>* Step 1:</p> <p>Message: SetChargingProfileRequest</p> <ul style="list-style-type: none">- chargingProfile.chargingProfileKind must be <i>Dynamic</i>- chargingProfile. must be <i>TxDefaultProfile</i>- chargingProfile.chargingSchedule[0].chargingSchedulePeriods must be size 1- chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].startPeriod must be 0- chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].limit must be $6 * \text{<limit multiplier>}$ <p>Note: Check [csms_csKSmartChargingChargingProfileLimits] for <i><limit multiplier></i></p> <p>* Step 3:</p> <p>Message: UpdateDynamicScheduleRequest</p> <ul style="list-style-type: none">- chargingProfileId must be <i><SetChargingProfileRequest.chargingProfile.id from step 1></i>- scheduleUpdate.limit must be <i><not omitted></i> <p>* Step 5:</p> <p>Message: UpdateDynamicScheduleRequest</p> <ul style="list-style-type: none">- chargingProfileId must be <i><SetChargingProfileRequest.chargingProfile.id from step 1></i>- scheduleUpdate.limit must be <i><not omitted></i> <p>* Step 7:</p> <p>Message: ClearChargingProfileRequest</p> <ul style="list-style-type: none">- chargingProfileId <i><chargingProfileId from step 1 chargingProfile></i>- chargingProfileCriteria <i>omit</i>
<p>Post scenario validations:</p> <p>N/a</p>

TC_Q_110_CSMS: Central V2X control with dynamic CSMS setpoint - pull

Test case name	Central V2X control with dynamic CSMS setpoint - pull
Test case Id	TC_Q_110_CSMS
Use case Id(s)	Q04, K28
Requirement(s)	K28.FR.01, K28.FR.02, K28.FR.07, K28.FR.12
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 support DynamicControl charging profiles and a CS is able to pull from CSMS.
Prerequisite(s)	- CSMS supports Bidirectional Power Transfer

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 send a TxDefaultProfile dynamic charging profile to the CS with - dynUpdateInterval 15	
2. The Test System responds with a SetChargingProfileResponse with status Accepted	1. The CSMS sends a SetChargingProfileRequest
3. The Test System sends a PullDynamicScheduleUpdateRequest with chargingProfileId <unknown chargingProfileId>	4. The CSMS responds with a PullDynamicScheduleUpdateResponse
5. The Test System sends a PullDynamicScheduleUpdateRequest with chargingProfileId <SetChargingProfileRequest.chargingProfile.id from step 1>	6. The CSMS responds with a PullDynamicScheduleUpdateResponse

Tool validations
<p>* Step 1:</p> <p>Message: SetChargingProfileRequest</p> <ul style="list-style-type: none">- chargingProfile.chargingProfileKind must be <i>Dynamic</i>- chargingProfile.dynUpdateInterval must be <i>15</i>- chargingProfile.dynUpdateTime must be <i><not omitted></i>- chargingProfile.chargingSchedule must be size = <i>1</i>- chargingProfile.chargingSchedule[0].chargingSchedulePeriods must be size = <i>1</i>- chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].startPeriod must be <i>0</i> <p>* Step 4:</p> <p>Message: PullDynamicScheduleUpdateResponse</p> <ul style="list-style-type: none">- status must be <i>Rejected</i> <p>* Step 6:</p> <p>Message: PullDynamicScheduleUpdateResponse</p> <ul style="list-style-type: none">- status must be <i>Accepted</i>- scheduleUpdate must be <i><not omitted></i>- scheduleUpdate must at least have a single field set
<p>Post scenario validations:</p> <p>N/a</p>

TC_Q_111_CSMS: External V2X control - with a charging profile from CSMS - setpoint

Test case name	External V2X control - with a charging profile from CSMS - setpoint
Test case Id	TC_Q_111_CSMS
Use case Id(s)	Q05
Requirement(s)	...
System under test	CSMS
Description	CSMS explicitly allows an External System to control the charge and discharge behaviour of an EV for a certain period of time.
Purpose	To verify if the CSMS is able to configure the CS to use an External V2X setpoint with relative or dynamic charging profile.
Prerequisite(s)	- CSMS supports Bidirectional Power Transfer

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 CSMS to set a TxDefaultProfile, Relative Charging Profile with chargingSchedulePeriod with operationMode ExternalSetpoint	
2. The Test System responds with a SetChargingProfileResponse with status Accepted	1. The CSMS sends a SetChargingProfileRequest
<u>Manual action:</u> trigger CSMS to set a TxDefaultProfile, Dynamic Charging Profile with chargingSchedulePeriod with operationMode ExternalSetpoint	
4. The Test System responds with a SetChargingProfileResponse with status Accepted	3. The CSMS sends a SetChargingProfileRequest

Tool validations
<p>* Step 1: Message: SetChargingProfileRequest</p> <ul style="list-style-type: none">- evseld must be <i><Configured evseld></i>- chargingProfile.chargingProfilePurpose must be <i>TxDefaultProfile</i>- chargingProfile.chargingProfileKind <i>Relative</i>- chargingProfile.dynUpdateTime must be <i><omitted></i>- chargingProfile.chargingSchedule[*].chargingSchedulePeriod[*].operationMode must be <i>ExternalSetpoint</i>- chargingProfile.chargingSchedule[*].chargingSchedulePeriod[*].setpoint must be <i><not omitted></i> <p>* Step 3: Message: SetChargingProfileRequest</p> <ul style="list-style-type: none">- evseld must be <i><Configured evseld></i>- chargingProfile.chargingProfilePurpose must be <i>TxDefaultProfile</i>- chargingProfile.chargingProfileKind <i>Dynamic</i>- chargingProfile.dynUpdateTime must be <i><Datetime of moment of step 3></i>- chargingProfile.chargingSchedule[*].chargingSchedulePeriod[*].operationMode must be <i>ExternalSetpoint</i>- chargingProfile.chargingSchedule[*].chargingSchedulePeriod[*].setpoint must be <i><not omitted></i>- chargingProfile.chargingSchedule[0].chargingSchedulePeriod size must be <i>1</i>- chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].startPeriod must be <i>0</i> <p>Post scenario validations:</p> <p>N/a</p>

TC_Q_112_CSMS: External V2X control - with a charging profile from CSMS - limit

Test case name	External V2X control - with a charging profile from CSMS - limit
Test case Id	TC_Q_112_CSMS
Use case Id(s)	Q05
Requirement(s)	...
System under test	CSMS
Description	CSMS explicitly allows an External System to control the charge and discharge behaviour of an EV for a certain period of time.
Purpose	To verify if the CSMS is able to configure the CS to use an External V2X limit.
Prerequisite(s)	- CSMS supports Bidirectional Power Transfer

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 CSMS to set a TxDefaultProfile, Relative Charging Profile with chargingSchedulePeriod with operationMode ExternalLimit	
2. The Test System responds with a SetChargingProfileResponse with status Accepted	1. The CSMS sends a SetChargingProfileRequest
<u>Manual action:</u> trigger CSMS to set a TxDefaultProfile, Dynamic Charging Profile with chargingSchedulePeriod with operationMode ExternalLimit	
4. The Test System responds with a SetChargingProfileResponse with status Accepted	3. The CSMS sends a SetChargingProfileRequest

Tool validations
<div><div>* Step 1: Message: SetChargingProfileRequest</div><div><div>- evseld must be <i><Configured evseld></i></div><div>- chargingProfile.chargingProfilePurpose must be <i>TxDefaultProfile</i></div><div>- chargingProfile.chargingProfileKind <i>Relative</i></div><div>- chargingProfile.dynUpdateTime must be <i><omitted></i></div><div>- chargingProfile.chargingSchedule[*].chargingSchedulePeriod[*].operationMode must be <i>ExternalLimits</i></div><div>- chargingProfile.chargingSchedule[*].chargingSchedulePeriod[*].limit must be <i><not omitted></i></div><div>- chargingProfile.chargingSchedule[*].chargingSchedulePeriod[*].setpoint must be <i><omitted></i></div></div></div> <div><div>* Step 3: Message: SetChargingProfileRequest</div><div><div>- evseld must be <i><Configured evseld></i></div><div>- chargingProfile[0].chargingProfilePurpose must be <i>TxDefaultProfile</i></div><div>- chargingProfile.chargingProfileKind <i>Dynamic</i></div><div>- chargingProfile.dynUpdateTime must be <i><Datetime of moment of step 3></i></div><div>- chargingProfile.chargingSchedule[*].chargingSchedulePeriod[*].operationMode must be <i>ExternalLimits</i></div><div>- chargingProfile.chargingSchedule[*].chargingSchedulePeriod[*].limit must be <i><not omitted></i> _</div><div>- chargingProfile.chargingSchedule[*].chargingSchedulePeriod[*].setpoint must be <i><omitted></i> _</div><div>- chargingProfile.chargingSchedule[0].chargingSchedulePeriod size must be <i>1</i></div><div>- chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].startPeriod must be <i>0</i></div></div></div>
<div>Post scenario validations:</div> <div>N/a</div>

TC_Q_117_CSMS: Frequency Support - Central V2X control - push

Test case name	Frequency Support - Central V2X control - push
Test case Id	TC_Q_117_CSMS
Use case Id(s)	Q07, Q04, K28
Requirement(s)	K28.FR.01, K28.FR.02
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 support DynamicControl charging profiles and push changes to the CS.
Prerequisite(s)	- CSMS supports Bidirectional Power Transfer

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 send a TxDefaultProfile dynamic charging profile to the CS and update the schedule every 15 seconds with - operationMode CentralFrequency - setpoint -6 * <limit multiplier> Note: Check [csms_csKSmartChargingChargingProfileLimits] for <limit multiplier>	
2. The Test System responds with a SetChargingProfileResponse with status Accepted	1. The CSMS sends a SetChargingProfileRequest
4. The Test System responds with a UpdateDynamicScheduleResponse with status Accepted	3. The CSMS sends a UpdateDynamicScheduleRequest

Tool validations
<p>* Step 1:</p> <p>Message: SetChargingProfileRequest</p> <ul style="list-style-type: none"> - chargingProfile.chargingProfileKind must be Dynamic - chargingProfile. must be TxDefaultProfile - chargingProfile.chargingSchedule[0].chargingSchedulePeriods must be size 1 - chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].startPeriod must be 0 - chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].setpoint must be $-6 * \langle \text{limit multiplier} \rangle$ <p>Note: Check [csms_csKSmartChargingChargingProfileLimits] for $\langle \text{limit multiplier} \rangle$</p> <p>* Step 3:</p> <p>Message: UpdateDynamicScheduleRequest</p> <ul style="list-style-type: none"> - chargingProfileId must be <SetChargingProfileRequest.chargingProfile.id from step 1> - scheduleUpdate.limit must be <not omitted>
Post scenario validations: N/a

TC_Q_120_CSMS: Frequency Support - Local V2X control - AFRR support

Test case name	Frequency Support - Local V2X control - AFRR support
Test case Id	TC_Q_120_CSMS
Use case Id(s)	Q08
Requirement(s)	Q08.FR.01, Q08.FR.02, Q08.FR.05, Q08.FR.12
System under test	CSMS
Description	To allow an EV to be used for aFRR frequency control, depending on a aFRR signal request that CSMS receives from an external actor.
Purpose	To verify if the CSMS is able to produce a charging profile for aFRR frequency support.
Prerequisite(s)	<ul style="list-style-type: none"> - CSMS supports Bidirectional Power Transfer - CSMS has Frequency support

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 CSMS to set a TxDefaultProfile Charging Profile with chargingSchedulePeriod with operationMode LocalFrequency and v2xBaseline and v2xFreqWattCurve and v2xSignalWattCurve.	
2. The Test System responds with a SetChargingProfileResponse with status Accepted	1. The CSMS the sends a SetChargingProfileRequest
<u>Manual action:</u> trigger CSMS to send out a AFRRSignalRequest with a given value for signal	
4. The Test System responds with a AFRRSignalResponse with status Accepted	3. The CSMS sends a AFRRSignalRequest

Tool validations

* Step 1:

Message: **SetChargingProfileRequest**

- **chargingProfile.chargingProfilePurpose** must be *TxDefaultProfile*
- **chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].operationMode** must be *LocalFrequency*
- **chargingProfile.chargingSchedule[0].chargingRateUnit** must be *W*
- **chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].limit** must be *<omitted>*
- **chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].limit_L2** must be *<omitted>*
- **chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].limit_L3** must be *<omitted>*

- **chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].dischargeLimit** must be *<omitted>*
- **chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].dischargeLimit_L2** must be *<omitted>*
- **chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].dischargeLimit_L3** must be *<omitted>*

- **chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].setpoint** must be *<omitted>*
- **chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].setpoint_L2** must be *<omitted>*
- **chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].setpoint_L3** must be *<omitted>*

- **chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].setpointReactive** must be *<omitted>*
- **chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].setpointReactive_L2** must be *<omitted>*
- **chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].setpointReactive_L3** must be *<omitted>*

- **chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].v2xBaseline** must be *<not omitted>*

- **chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].v2xFreqWattCurve** size must be *>= 2*
- **chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].v2xFreqWattCurve[*].frequency** must be *<not omitted>*
- **chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].v2xFreqWattCurve[*].power** must be *<not omitted>*

- **chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].v2xSignalWattCurve** size must be *>= 2*
- **chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].v2xSignalWattCurve[*].signal** must be *<not omitted>*
- **chargingProfile.chargingSchedule[0].chargingSchedulePeriod[0].v2xSignalWattCurve[*].power** must be *<not omitted>*

* Step 3:

Message: **AFRRSignalRequest**

- **signal** must *<be the value that was provided in manual action before step 3.>*

Post scenario validations:

N/a

TC_Q_121_CSMS: Frequency Support - Local V2X control

Test case name	Frequency Support - Local V2X control
Test case Id	TC_Q_121_CSMS
Use case Id(s)	Q08
Requirement(s)	Q08.FR.01, Q08.FR.02, Q08.FR.05
System under test	CSMS
Description	To allow an EV to be used for frequency control, depending on local frequency readings.
Purpose	To check if the CSMS supports Local V2X control for frequency support.
Prerequisite(s)	- CSMS supports Bidirectional Power Transfer - CSMS has Frequency support

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is ISO15118-20 transaction just started exposing: - <allowedEnergyTransferModes> - <transactionId>

Main (Test scenario)	
Charging Station	CSMS
Agree on energy transfer mode	
1. The Test System sends a NotifyEVChargingNeedsRequest with evseld = <Configured evseld> chargingNeeds.requestedEnergyTransfer AC_BPT chargingNeeds.availableEnergyTransfer [AC_single_phase, AC_two_phase, AC_three_phase, AC_BPT, AC_BPT_DER, AC_DER] chargingNeeds.controlMode DynamicControl chargingNeeds.v2xChargingParameters.maxChargePower 1234W chargingNeeds.v2xChargingParameters.maxDischargePower 4321W	2. The CSMS responds with a NotifyEVChargingNeedsResponse
4. And Test System responds with a SetChargingProfileResponse with status Accepted	3. The CSMS sends a SetChargingProfileRequest
<u>Manual Action:</u> Request the CSMS to send a LocalFrequency charging profile to the CS - evseld <Configured evseld> - chargingProfile.transactionId <transactionId> - chargingProfile.chargingProfilePurpose TxProfile - chargingProfile.chargingSchedule.chargingRateUnit W - chargingProfile.chargingSchedule.chargingSchedulePeriod.operationMode must be LocalFrequency - chargingProfile.chargingSchedule.chargingSchedulePeriod.v2xFreqWattCurve[0].frequency 49.5 - chargingProfile.chargingSchedule.chargingSchedulePeriod.v2xFreqWattCurve[0].power -1000 - chargingProfile.chargingSchedule.chargingSchedulePeriod.v2xFreqWattCurve[1].frequency 50.0 - chargingProfile.chargingSchedule.chargingSchedulePeriod.v2xFreqWattCurve[1].power 0 - chargingProfile.chargingSchedule.chargingSchedulePeriod.v2xFreqWattCurve[2].frequency 50.5 - chargingProfile.chargingSchedule.chargingSchedulePeriod.v2xFreqWattCurve[2].power 1000 - chargingProfile.chargingSchedule.chargingSchedulePeriod.v2xBaseline 7000	

Main (Test scenario)	
6. And Test System responds with a SetChargingProfileResponse with status <i>Accepted</i>	5. The CSMS sends a SetChargingProfileRequest

Tool validations
<p>* Step 2: Message: NotifyEVChargingNeedsResponse - status must be <i>Accepted</i> or <i>Processing</i></p> <p>* Step 5: Message: SetChargingProfileRequest - evseld must be <i><Configured evseld></i> - chargingProfile.transactionId must be <i><transactionId></i> - chargingProfile.chargingProfilePurpose must be <i>TxProfile</i> - chargingProfile.chargingSchedule.chargingRateUnit must be <i>W</i> - chargingProfile.chargingSchedule.chargingSchedulePeriod.operationMode must be <i>LocalFrequency</i> - chargingProfile.chargingSchedule.chargingSchedulePeriod.limit must be <i><omitted></i> - chargingProfile.chargingSchedule.chargingSchedulePeriod.limit_L2 must be <i><omitted></i> - chargingProfile.chargingSchedule.chargingSchedulePeriod.limit_L3 must be <i><omitted></i> - chargingProfile.chargingSchedule.chargingSchedulePeriod.dischargeLimit must be <i><omitted></i> - chargingProfile.chargingSchedule.chargingSchedulePeriod.dischargeLimit_L2 must be <i><omitted></i> - chargingProfile.chargingSchedule.chargingSchedulePeriod.dischargeLimit_L3 must be <i><omitted></i> - chargingProfile.chargingSchedule.chargingSchedulePeriod.setpoint must be <i><omitted></i> - chargingProfile.chargingSchedule.chargingSchedulePeriod.setpoint_L2 must be <i><omitted></i> - chargingProfile.chargingSchedule.chargingSchedulePeriod.setpoint_L3 must be <i><omitted></i> - chargingProfile.chargingSchedule.chargingSchedulePeriod.setpointReactive must be <i><omitted></i> - chargingProfile.chargingSchedule.chargingSchedulePeriod.setpointReactive_L2 must be <i><omitted></i> - chargingProfile.chargingSchedule.chargingSchedulePeriod.setpointReactive_L3 must be <i><omitted></i> - chargingProfile.chargingSchedule.chargingSchedulePeriod.v2xFreqWattCurve[0].frequency must be <i><not omitted></i> - chargingProfile.chargingSchedule.chargingSchedulePeriod.v2xFreqWattCurve[0].power must be <i><not omitted></i> - chargingProfile.chargingSchedule.chargingSchedulePeriod.v2xFreqWattCurve[1].frequency must be <i><not omitted></i> - chargingProfile.chargingSchedule.chargingSchedulePeriod.v2xFreqWattCurve[1].power must be <i><not omitted></i> - chargingProfile.chargingSchedule.chargingSchedulePeriod.v2xFreqWattCurve[2].frequency must be <i><not omitted></i> - chargingProfile.chargingSchedule.chargingSchedulePeriod.v2xFreqWattCurve[2].power must be <i><not omitted></i> - chargingProfile.chargingSchedule.chargingSchedulePeriod.v2xBaseline must be <i><not omitted></i></p>
<p>Post scenario validations: N/a</p>

TC_Q_124_CSMS: Local V2X control for load balancing - good flow

Test case name	Local V2X control for load balancing - good flow
Test case Id	TC_Q_124_CSMS
Use case Id(s)	Q09
Requirement(s)	...
System under test	CSMS
Description	To allow the EV to be utilized for locally controlled load balancing.
Purpose	To verify if the CSMS is able to configure a CS for locally controlled load balancing
Prerequisite(s)	- CSMS supports Bidirectional Power Transfer

Before (Preparations)
Configuration State: - CS reports V2XChargingCtrlr.SupportedOperationModes that contains <i>LocalLoadBalancing</i>
Memory State: N/a
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
<u>Manual action:</u> trigger CSMS to set local load-balancing thresholds: - <i>UpperThreshold = 1000, LowerThreshold = -1000</i> - <i>UpperOffset = 200, LowerOffset = 0</i>	
2. The Test System responds with a SetVariablesResponse with setVariableResult[0].component.name <i>V2XChargingCtrlr</i> setVariableResult[0].component.evse.id <i>1</i> setVariableResult[0].variable.name <i>V2XLocalLoadBalancing</i> setVariableResult[0].variable.instance <i>UpperThreshold</i> setVariableResult[0].attributeStatus <i>Accepted</i> setVariableResult[1].component.name <i>V2XChargingCtrlr</i> setVariableResult[1].component.evse.id <i>1</i> setVariableResult[1].variable.name <i>V2XLocalLoadBalancing</i> setVariableResult[1].variable.instance <i>LowerThreshold</i> setVariableResult[1].attributeStatus <i>Accepted</i> setVariableResult[2].component.name <i>V2XChargingCtrlr</i> setVariableResult[2].component.evse.id <i>1</i> setVariableResult[2].variable.name <i>V2XLocalLoadBalancing</i> setVariableResult[2].variable.instance <i>UpperOffset</i> setVariableResult[2].attributeStatus <i>Accepted</i> setVariableResult[3].component.name <i>V2XChargingCtrlr</i> setVariableResult[3].component.evse.id <i>1</i> setVariableResult[3].variable.name <i>V2XLocalLoadBalancing</i> setVariableResult[3].variable.instance <i>LowerOffset</i> setVariableResult[3].attributeStatus <i>Accepted</i>	1. The CSMS sends a SetVariablesRequest <u>Note:</u> CSMS may send these variables in one or multiple requests
CSMS sets a Charging Profile with <i>LocalLoadBalancing</i>	
<u>Manual action:</u> trigger CSMS to set a TxDefaultProfile Charging Profile with <i>chargingSchedulePeriod</i> with <i>operationMode LocalLoadBalancing</i>	
4. The Test System responds with a SetChargingProfileResponse with status <i>Accepted</i>	3. The CSMS the sends a SetChargingProfileRequest

Tool validations
<p>* Step 1:</p> <p>Message SetVariablesRequest with</p> <p>setVariableData[0].component.name <i>V2XChargingCtrlr</i></p> <p>setVariableData[0].component.evse.id <i>1</i></p> <p>setVariableData[0].variable.name <i>V2XLocalLoadBalancing</i></p> <p>setVariableData[0].variable.instance <i>UpperThreshold</i></p> <p>setVariableData[0].attributeValue <i>2000</i></p> <p>setVariableData[1].component.name <i>V2XChargingCtrlr</i></p> <p>setVariableData[1].component.evse.id <i>1</i></p> <p>setVariableData[1].variable.name <i>V2XLocalLoadBalancing</i></p> <p>setVariableData[1].variable.instance <i>LowerThreshold</i></p> <p>setVariableData[1].attributeValue <i>-1000</i></p> <p>setVariableData[2].component.name <i>V2XChargingCtrlr</i></p> <p>setVariableData[2].component.evse.id <i>1</i></p> <p>setVariableData[2].variable.name <i>V2XLocalLoadBalancing</i></p> <p>setVariableData[2].variable.instance <i>UpperOffset</i></p> <p>setVariableData[2].attributeValue <i>200</i></p> <p>setVariableData[3].component.name <i>V2XChargingCtrlr</i></p> <p>setVariableData[3].component.evse.id <i>1</i></p> <p>setVariableData[3].variable.name <i>V2XLocalLoadBalancing</i></p> <p>setVariableData[3].variable.instance <i>LowerOffset</i></p> <p>setVariableData[3].attributeValue <i>0</i></p> <p><u>Note</u>: Alternatively these may be sent as individual requests</p> <p>* Step 3:</p> <p>Message: SetChargingProfileRequest</p> <ul style="list-style-type: none">- evseld must be <i><Configured evseld></i>- chargingProfile.chargingSchedule[*].chargingSchedulePeriod[*].operationMode must be <i>LocalLoadBalancing</i>- chargingProfile.chargingSchedule[*].chargingSchedulePeriod[*].limit(_L2/L3) must be <i>_<omitted>_</i>- chargingProfile.chargingSchedule[*].chargingSchedulePeriod[*].dischargingLimit(_L2/L3) must be <i>_<omitted>_</i>- chargingProfile.chargingSchedule[*].chargingSchedulePeriod[*].setpoint(_L2/L3) must be <i>_<omitted>_</i>- chargingProfile.chargingSchedule[*].chargingSchedulePeriod[*].setpointReactive(_L2/L3) must be <i>_<omitted>_</i> <p>Post scenario validations:</p> <p>N/a</p>

TC_Q_125_CSMS: Idle operationMode - Idle, minimizing energy consumption - Idle with EvseSleep

Test case name	Idle operationMode - Idle, minimizing energy consumption - Idle with EvseSleep
Test case Id	TC_Q_150_CS
Use case Id(s)	Q10
Requirement(s)	Q10.FR.01
System under test	CSMS
Description	To request the EV to not perform any charging or discharging. Preconditioning of the vehicle is allowed.
Purpose	To verify that the CSMS is able to send a Charging Profile which Minimizes the energy consumption.
Prerequisite(s)	<ul style="list-style-type: none"> - CSMS supports ISO15118-20 - CS supports V2XChargingCtrlr.V2XSupportedOperationModes containing Idle - CS supports SmartChargingCtrlr.SupportsFeature[EvseSleep] = 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 <i>ISO15118-20 transaction started in V2X AC_BPT energy transfer mode</i>	
<i>Without EVSE sleeping</i>	
<u>Manual action:</u> trigger CSMS to set a TxProfile Charging Profile with chargingSchedulePeriod with operationMode Idle and evseSleep false	
3. The Test System responds with a SetChargingProfileResponse with status must be <i>Accepted</i>	2. The CSMS the sends a SetChargingProfileRequest
4. The Test System sends a TransactionEventRequest with eventType <i>Updated</i> triggerReason <i>OperationModeChanged</i> transactionInfo.transactionId <transactionId> transactionInfo.operationMode <i>Idle</i> evseSleep <i>false</i>	5. The CSMS responds with a TransactionEventResponse
<i>With EVSE sleeping</i>	
<u>Manual action:</u> trigger CSMS to set a TxProfile Charging Profile with chargingSchedulePeriod with operationMode Idle and evseSleep true	
7. The Test System responds with a SetChargingProfileResponse	6. The CSMS the sends a SetChargingProfileRequest
8. The Test System sends a TransactionEventRequest with eventType <i>Updated</i> triggerReason <i>MeterValuePeriodic</i> transactionInfo.transactionId <transactionId> transactionInfo.operationMode <i>Idle</i> evseSleep <i>true</i>	9. The CSMS responds with a TransactionEventResponse

Tool validations
<p>* Step 2:</p> <p>Message: SetChargingProfileRequest</p> <ul style="list-style-type: none"> - chargingProfile.chargingProfilePurpose must be <i>TxProfile</i> - chargingProfile.chargingSchedule.chargingSchedulePeriod.operationMode must be <i>Idle</i> - chargingProfile.chargingSchedule.chargingSchedulePeriod.evseSleep must be <i>false</i> - chargingProfile.chargingSchedule.chargingSchedulePeriod.limit must be <i><omitted></i> - chargingProfile.chargingSchedule.chargingSchedulePeriod.dischargingLimit must be <i><omitted></i> - chargingProfile.chargingSchedule.chargingSchedulePeriod.setpoint must be <i><omitted></i> - chargingProfile.chargingSchedule.chargingSchedulePeriod.setpointReactive must be <i><omitted></i> <p>* Step 6:</p> <p>Message: SetChargingProfileRequest</p> <ul style="list-style-type: none"> - chargingProfile.chargingProfilePurpose must be <i>TxProfile</i> - chargingProfile.chargingSchedule.chargingSchedulePeriod.operationMode must be <i>Idle</i> - chargingProfile.chargingSchedule.chargingSchedulePeriod.evseSleep must be <i>true</i> - chargingProfile.chargingSchedule.chargingSchedulePeriod.limit must be <i><omitted></i> - chargingProfile.chargingSchedule.chargingSchedulePeriod.dischargingLimit must be <i><omitted></i> - chargingProfile.chargingSchedule.chargingSchedulePeriod.setpoint must be <i><omitted></i> - chargingProfile.chargingSchedule.chargingSchedulePeriod.setpointReactive must be <i><omitted></i>
<p>Post scenario validations:</p> <p>N/a</p>

R DER Control

TC_R_107_CSMS: Configure DER control settings at CS

Test case name	Configure DER control settings at CS
Test case Id	TC_R_107_CSMS
Use case Id(s)	R04
Requirement(s)	R04.FR.12, R04.FR.14
System under test	CSMS
Description	Setting some DER controls and reboot the charging station. After reboot, retrieving the configured DER controls.
Purpose	To check if the CSMS is able to set DER Controls.
Prerequisite(s)	CSMS supports DER Control The Test System must report support for the following; component.name <i>ACDERCtrlr</i> variable.name <i>ModesSupported</i> variableAttribute.value [<i>FreqWatt, FreqDroop, LimitMaxDischarge, Gradients, EnterService</i>]

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

Main (Test scenario)	
Charging Station	CSMS
Test ClearDERControl	
Manual action: trigger CSMS to send a ClearDERControlRequest with isDefault true	
2. The Test System responds with a ClearDERControlResponse with status <i>Accepted</i>	1. The CSMS sends a ClearDERControlRequest
Manual action: trigger CSMS to send a ClearDERControlRequest with isDefault false	
4. The Test System responds with a ClearDERControlResponse with status <i>Accepted</i>	3. The CSMS sends a ClearDERControlRequest
Test SetDERControl	
Manual action: trigger CSMS to send a SetDERControlRequest with isDefault true, controlType FreqDroop with priority 6	
6. The Test System responds with a SetDERControlResponse with status <i>Accepted</i>	5. The CSMS sends a SetDERControlRequest
Manual action: trigger CSMS to send a SetDERControlRequest with isDefault false controlType FreqWatt with a startTime, duration of 900 and priority 4	
8. The Test System responds with a SetDERControlResponse with status <i>Accepted</i>	7. The CSMS sends a SetDERControlRequest
Manual action: trigger CSMS to send a SetDERControlRequest for controlType EnterService with priority = 1	

Main (Test scenario)	
<p>10. The Test System responds with a SetDERControlResponse with status <i>Accepted</i></p> <p><i>Test GetDERControl</i></p>	<p>9. The CSMS sends a SetDERControlRequest</p>
<u>Manual action</u> : trigger CSMS to send a <i>GetDERControlRequest</i> with <i>isDefault true</i>	
<p>12. The Test System responds with a GetDERControlResponse with status <i>Accepted</i></p>	<p>11. The CSMS sends a GetDERControlRequest</p>
<p>13. The Test System sends a ReportDERControlRequest with</p> <ul style="list-style-type: none"> - enterService : - enterService.id = "enterservice_1" - enterService.*enterService.highVoltage = 250 - enterService.*enterService.lowVoltage = 210 - enterService.*enterService.highFreq = 50.5 - enterService.*enterService.lowFreq 49.5 - enterService.*enterService.priority = 1 - freqDroop : - freqDroop.id = "freqdroop_1" - freqDroop.isDefault = true - freqDroop.isSuperseded = false - freqDroop.freqDroop.priority must be 6 - freqDroop.freqDroop.overFreq = 50.5 - freqDroop.freqDroop.underFreq = 49.5 - freqDroop.freqDroop.overDroop = 0.05 - freqDroop.freqDroop.underDroop = 0.05 - freqDroop.freqDroop.responseTime = 10 	<p>14. The CSMS sends a ReportDERControlResponse</p>
<u>Manual action</u> : trigger CSMS to send a <i>GetDERControlRequest</i> with <i>isDefault false</i>	
<p>16. The Test System responds with a GetDERControlResponse with status <i>Accepted</i></p>	<p>15. The CSMS sends a GetDERControlRequest</p>
<p>17. The Test System sends a ReportDERControlRequest with</p> <ul style="list-style-type: none"> - curve : - curve.id = "freqwatt_1" - curve.curveType = "FreqWatt" - curve.isDefault = false - curve.isSuperseded = false - curve.curve.priority = 4 - curve.curve.yUnit = "PctMaxW" - curve.curve.curvedata[0].x = 49 - curve.curve.curvedata[0].y = 75 - curve.curve.curvedata[1].x = 49.5 - curve.curve.curvedata[1].y = 90 - curve.curve.curvedata[2].x = 50.5 - curve.curve.curvedata[2].y = 100 - curve.curve.curvedata[3].x = 51 - curve.curve.curvedata[3].y = 100 - curve.curve.startTime = <current time> - curve.curve.duration = 900 	<p>18. The CSMS sends a ReportDERControlResponse</p>
<u>Manual action</u> : trigger CSMS to send a <i>GetDERControlRequest</i> with <i>controlType LimitMaxDischarge</i>	

Main (Test scenario)	
20. The Test System responds with a GetDERControlResponse with status <i>Accepted</i>	19. The CSMS sends a GetDERControlRequest
<u>Manual action:</u> <i>trigger CSMS to send a GetDERControlRequest with a controlId</i>	
22. The Test System responds with a GetDERControlResponse with status <i>Accepted</i>	21. The CSMS sends a GetDERControlRequest
23. The Test System sends a NotifyDERStartStopRequest with controlId <i><SetDERControlRequest.controlId of step 11></i> started <i>true</i> timestamp <i><Current date/time></i> supersededIds <i>[<SetDERControlRequest.controlId of step 9>]</i>	24. The CSMS responds with a NotifyDERStartStopResponse
Tool validations	
<p>* Step 1: Message: ClearDERControlRequest - isDefault <i>true</i></p> <p>* Step 3: Message: ClearDERControlRequest - isDefault <i>false</i></p> <p>* Step 5: Message: SetDERControlRequest - isDefault must be <i>true</i> - controlType must be <i>FreqDroop</i> - freqDroop.priority must be <i>6</i> - freqDroop.startTime must be <i><omitted></i> - freqDroop.duration must be <i><omitted></i> - <i>Other optional fields must be omitted</i></p>	

Tool validations
<p>* Step 7:</p> <p>Message: SetDERControlRequest</p> <ul style="list-style-type: none">- isDefault must be <i>false</i>- controlType must be <i>FreqWatt</i>- curve.priority must be <i>4</i>- curve.startTime must be <i><not omitted></i>- curve.duration must be <i>900</i>- curve.curveData[*].x must be <i><not omitted></i>- curve.curveData[*].y must be <i><not omitted></i> - <i>Other optional fields must be omitted</i>
<p>* Step 9:</p> <p>Message: SetDERControlRequest</p> <ul style="list-style-type: none">- isDefault must be <i>false</i>- controlType must be <i>EnterService</i>- enterService must be <i><not omitted></i>- enterService.priority must be <i>1</i> - <i>Other optional fields must be omitted</i>
<p>* Step 11:</p> <p>Message: GetDERControlRequest</p> <ul style="list-style-type: none">- isDefault must be <i>true</i>
<p>* Step 15:</p> <p>Message: GetDERControlRequest</p> <ul style="list-style-type: none">- isDefault must be <i>false</i>
<p>* Step 19:</p> <p>Message: GetDERControlRequest</p> <ul style="list-style-type: none">- controlType must be <i>VoltVar</i>
<p>* Step 21:</p> <p>Message: GetDERControlRequest</p> <ul style="list-style-type: none">- controlId must be <i><not omitted></i>
<p>Post scenario validations:</p> <p>N/a</p>

TC_R_108_CSMS: Charging station reporting a DER event

Test case name	Charging station reporting a DER event
Test case Id	TC_R_108_CSMS
Use case Id(s)	R05
Requirement(s)	
System under test	CSMS
Description	Configure a DER Control which will always trigger and halves the charging rate of the charging profile when a transaction starts, then add another DER Control to test overriding the previous one. After the clearing all DER controls to test it reports the alert should be cleared.
Purpose	To check if the CSMS supports receiving DER events.
Prerequisite(s)	CSMS supports DER Control

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

Main (Test scenario)	
Charging Station	CSMS
Report DER alarm	
1. The Test System sends a NotifyDERAlarmRequest with controlType <i>LVMustTrip</i> gridEventFault <i>UnderVoltage</i> alarmEnded <i>false</i> timestamp <i><Current DateTime></i>	2. The CSMS responds with a NotifyDERAlarmResponse
<u>Note:</u> The Test System waits 5 seconds	
Clear DER alarm	
3. The Test System sends a NotifyDERAlarmRequest with controlType <i>LVMustTrip</i> gridEventFault <i>UnderVoltage</i> alarmEnded <i>true</i> timestamp <i><Current DateTime></i>	4. The CSMS responds with a NotifyDERAlarmResponse

Tool validations
N/a
Post scenario validations: N/a

S Battery Swapping

TC_S_102_CSMS: Battery Swap - Remote Start - not enough batteries

Test case name	Battery Swap - Remote Start - not enough batteries
Test case Id	TC_S_102_CSMS
Use case Id(s)	S01
Requirement(s)	
System under test	CSMS
Description	CSMS is able to support battery swapping.
Purpose	To verify whether the CSMS is able successful support battery swapping with not enough batteries.
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> Let CSMS trigger a request for swapping batteries	
2. The Test System responds with a RequestBatterySwapResponse with status <i>Rejected</i> statusInfo.reasonCode <i>NoBatteryAvailable</i>	1. The CSMS sends a RequestBatterySwapRequest

Tool validations
* Step 1: Message: RequestBatterySwapRequest - requestId must be <not omitted> - idToken.idToken must be <Configured valid_idtoken_idtoken>
Post scenario validations: N/a

TC_S_103_CSMS: Battery Swap - Remote Start - enough batteries available

Test case name	Battery Swap - Remote Start - enough batteries available
Test case Id	TC_S_103_CSMS
Use case Id(s)	S02, S03, S04
Requirement(s)	
System under test	CSMS
Description	CSMS is able to support battery swapping.
Purpose	To verify whether the CSMS is able successful support battery swapping.
Prerequisite(s)	Charging Station supports battery swapping. There are enough batteries available for swapping.

Before (Preparations)
Configuration State: BatterySwapCtrlr.Idtoken is <i><not set></i>
Memory State: N/a
Reusable State(s): N/a

Main (Test scenario)	
Charging Station	CSMS
<u>Manual Action:</u> Let CSMS trigger a request for swapping batteries	
2. The Test System responds with a RequestBatterySwapResponse with status <i>Accepted</i>	1. The CSMS sends a RequestBatterySwapRequest
<u>Note:</u> Set of two (virtual) batteries are inserted at this point in scenario.	
3. The Test System notifies the CSMS about the status change of the slot.	4. The CSMS responds accordingly.
5. The Test System sends a TransactionEventRequest with eventType <i>Started</i> triggerReason <i>CablePluggedIn</i> idToken.idToken "" idToken.type <i>NoAuthorization</i> evse <i><Configured evseld></i> evse.connectorId <i>1</i> transactionInfo.transactionId <i>111-222-333-444-3</i> transactionInfo.chargingState <i>EVConnected</i>	6. The CSMS responds with a TransactionEventResponse
7. The Test System notifies the CSMS about the status change of the slot.	8. The CSMS responds accordingly.
9. The Test System sends a TransactionEventRequest with eventType <i>Started</i> triggerReason <i>CablePluggedIn</i> idToken.idToken "" idToken.type <i>NoAuthorization</i> evse <i><Configured evseld2></i> evse.connectorId <i>1</i> transactionInfo.transactionId <i>111-222-333-444-4</i> transactionInfo.chargingState <i>EVConnected</i>	10. The CSMS responds with a TransactionEventResponse

Main (Test scenario)	
<p>11. The Test System sends a BatterySwapRequest with</p> <p>eventType <i>BatteryIn</i></p> <p>requestId <i><requestId></i></p> <p>idToken.idToken <i><Configured valid_idtoken_idtoken></i></p> <p>batteryData[0].evseld <i><Configured evseld></i>_</p> <p>batteryData[0].serialNumber <i>1234</i></p> <p>batteryData[0].soC <i>23</i></p> <p>batteryData[0].soH <i>85</i></p> <p>batteryData[1].evseld <i><Configured evseld2></i>_</p> <p>batteryData[1].serialNumber <i>5678</i></p> <p>batteryData[1].soC <i>45</i></p> <p>batteryData[1].soH <i>87</i></p>	<p>12. The CSMS responds with a BatterySwapResponse</p>
<p><u>Note:</u> Set of two batteries are extracted at this point in scenario.</p>	
<p>13. The Charging Station sends a TransactionEventRequest with</p> <p>eventType <i>Ended</i></p> <p>triggerReason <i>EnergyLimitReached</i></p> <p>idToken.idToken <i>""</i></p> <p>idToken.type <i>NoAuthorization</i></p> <p>transactionInfo.transactionId <i>111-222-333-444-1</i></p> <p>transactionInfo.chargingState <i>Idle</i></p> <p>transactionInfo.stoppedReason <i>EVDisconnected</i></p>	<p>14. The Test System responds with a TransactionEventResponse</p>
<p>15. The Test System notifies the CSMS about the status change of the slot.</p>	<p>16. The CSMS responds accordingly.</p>
<p>17. The Charging Station sends a TransactionEventRequest with</p> <p>eventType <i>Ended</i></p> <p>triggerReason <i>EnergyLimitReached</i></p> <p>idToken.idToken <i>""</i></p> <p>idToken.type <i>NoAuthorization</i></p> <p>transactionInfo.transactionId <i>111-222-333-444-2</i></p> <p>transactionInfo.chargingState <i>Idle</i></p> <p>transactionInfo.stoppedReason <i>EVDisconnected</i></p>	<p>18. The Test System responds with a TransactionEventResponse</p>
<p>19. The Test System notifies the CSMS about the status change of the slot.</p>	<p>20. The CSMS responds accordingly.</p>
<p>21. The Test System sends a BatterySwapRequest with</p> <p>eventType <i>BatteryOut</i></p> <p>requestId <i><requestId></i></p> <p>idToken.idToken <i><Configured valid_idtoken_idtoken></i></p> <p>batteryData[0].evseld <i><Configured evseld3></i>_</p> <p>batteryData[0].serialNumber <i>4321</i></p> <p>batteryData[0].soC <i>80</i></p> <p>batteryData[0].soH <i>95</i></p> <p>batteryData[1].evseld <i><Configured evseld4></i>_</p> <p>batteryData[1].serialNumber <i>8765</i></p> <p>batteryData[1].soC <i>85</i></p> <p>batteryData[1].soH <i>78</i></p>	<p>22. The CSMS responds with a BatterySwapResponse</p>
<p><u>Note:</u> Steps 13 and 14 are repeated for each battery taken out</p>	

Tool validations
<p>* Step 1: Message: RequestBatterySwapResponse - requestId must be <i><not omitted></i> - idToken.idToken must be <i><Configured valid_idtoken_idtoken></i></p> <p>* Step 6: Message: TransactionEventResponse - idTokenInfo.status must be <i>Accepted</i></p> <p>* Step 10: Message: TransactionEventResponse - idTokenInfo.status must be <i>Accepted</i></p> <p>* Step 12: Message: BatterySwapResponse</p> <p>* Step 14: Message: TransactionEventResponse - idTokenInfo.status must be <i>Accepted</i></p> <p>* Step 18: Message: TransactionEventResponse - idTokenInfo.status must be <i>Accepted</i></p> <p>* Step 22: Message: BatterySwapResponse</p>
<p>Post scenario validations: N/a</p>

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.

Booted

State	Booted
System under test	CSMS
Description	This state will simulate that the Charging Station is completely power cycled. The Test System 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 (Test scenario)	
Charging Station	CSMS
1. The Test System sends a BootNotificationRequest with reason <i>PowerUp</i> chargingStation.model <Configured model> chargingStation.vendorName <Configured vendorName>	2. The CSMS responds with a BootNotificationResponse
3. The Test System notifies the CSMS about the current state of all connectors. Message: StatusNotificationRequest with connectorStatus <i>Available</i> Message: NotifyEventRequest with trigger <i>Delta</i> actualValue <i>"Available"</i> component.name <i>"Connector"</i> variable.name <i>"AvailabilityState"</i>	4. The CSMS responds accordingly.

Tool validations
* Step 2: Message: BootNotificationResponse - status <i>Accepted</i>
Post scenario validations: State is <i>Booted</i>

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 (Test scenario)	
Charging Station	CSMS
Manual Action: <i>Trigger the CSMS to send a ReserveNowRequest for specific EVSE.</i>	
2. The Test System responds with a ReserveNowResponse With status <i>Accepted</i>	1. The CSMS sends a ReserveNowRequest
3. The Test System notifies the CSMS about the current state of the connector(s) of the Specified EVSE Message: StatusNotificationRequest with connectorStatus <i>Reserved</i> Message: NotifyEventRequest with trigger <i>Delta</i> actualValue <i>"Reserved"</i> component.name <i>"Connector"</i> variable.name <i>"AvailabilityState"</i>	4. The CSMS responds accordingly.

Tool validations
* Step 1: Message: ReserveNowRequest - evseld must be <i><Specified evseld></i> - connectorType must be omitted - idToken.idToken <i><Configured valid_idtoken_idtoken></i> - idToken.type <i><Configured valid_idtoken_type></i>
Post scenario validations: State is <i>Reserved</i>

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 (Test scenario)	
Charging Station	CSMS
Manual Action: Request the CSMS to change the availability of the specified components to Inoperative.	
2. The Test System responds with a ChangeAvailabilityResponse with status <i>Accepted</i>	1. The CSMS sends a ChangeAvailabilityRequest
3. The Test System notifies the CSMS about the current state of all connectors belonging to the specified EVSE (and optionally also from the EVSE itself). Message: StatusNotificationRequest - connectorStatus <i>Unavailable</i> Message: NotifyEventRequest - trigger <i>Delta</i> - actualValue <i>"Unavailable"</i> - component.name <i>"ChargingStation" / EVSE / Connector</i> - variable.name <i>"AvailabilityState"</i>	4. The CSMS responds accordingly.

Tool validations
* Step 1: Message ChangeAvailabilityRequest - operationalStatus <i>Inoperative</i> - evse <i><Specified evseld></i> - connectorId <i>omitted</i>
Post scenario validations: State is <i>Unavailable</i>

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 (Test scenario)

Charging Station	CSMS
1. The Test System notifies the CSMS about the status change of the connector Message: StatusNotificationRequest - connectorStatus is <i>Occupied</i> Message: NotifyEventRequest - trigger is <i>Delta</i> - actualValue is <i>Occupied</i> - component.name is <i>Connector</i> - variable.name is <i>AvailabilityState</i>	2. The CSMS responds accordingly.
3. The Test System sends a TransactionEventRequest With triggerReason is <i>CablePluggedIn</i> transactionInfo.chargingState is <i>EVConnected</i> evse.id <Configured evseld> evse.connectorId <Configured connectorId> If State is <i>Authorized</i> then eventType is <i>Updated</i> else eventType is <i>Started</i>	4. The CSMS responds with a TransactionEventResponse

Tool validations

N/a

Post scenario validations:

State is *EVConnectedPreSession*

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 (Test scenario)	
Charging Station	CSMS
1. The Test System sends an <code>AuthorizeRequest</code> With <code>idToken.idToken</code> <Configured <i>valid_idtoken_idtoken</i> > <code>idToken.type</code> <Configured <i>valid_idtoken_type</i> >	2. The CSMS responds with an <code>AuthorizeResponse</code>
3. The Test System sends a <code>TransactionEventRequest</code> With <code>triggerReason</code> is <i>Authorized</i> <code>idToken.idToken</code> <Configured <i>valid_idtoken_idtoken</i> > <code>idToken.type</code> <Configured <i>valid_idtoken_type</i> > If <code>State</code> is <i>EVConnectedPreSession</i> then <code>eventType</code> is <i>Updated</i> else <code>eventType</code> is <i>Started</i>	4. The CSMS responds with a <code>TransactionEventResponse</code>

Tool validations
* Step 2: Message: <code>AuthorizeResponse</code> - <code>idTokenInfo.status</code> must be <i>Accepted</i> * Step 4: Message: <code>TransactionEventResponse</code> - <code>idTokenInfo.status</code> must be <i>Accepted</i>
Post scenario validations: <code>State</code> is <i>Authorized</i>

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) (Test scenario)

Charging Station	CSMS
1. The Test System notifies the CSMS about the status change of the connector. Message: StatusNotificationRequest - connectorStatus is <i>Occupied</i> Message: NotifyEventRequest - trigger is <i>Delta</i> - actualValue is <i>Occupied</i> - component.name is <i>Connector</i> - variable.name is <i>AvailabilityState</i>	2. The CSMS responds accordingly.
3. The Test System sends a TransactionEventRequest With triggerReason is <i>CablePluggedIn</i> transactionInfo.chargingState is <i>EVConnected</i> evse.id < <i>Configured evseld</i> > evse.connectorId < <i>Configured connectorId</i> > eventType is <i>Updated</i>	4. The CSMS responds with a TransactionEventResponse

Tool validations

N/a

Main (Part 2) (Test scenario)

Charging Station	CSMS
5. The Test System sends a TransactionEventRequest With triggerReason is <i>ChargingStateChanged</i> transactionInfo.chargingState is <i>Charging</i> eventType is <i>Updated</i>	6. The CSMS responds with a TransactionEventResponse

Tool validations

N/a

Post scenario validations:

State is *EnergyTransferStarted*

EVConnected is *true*

EnergyTransferSuspended

State	EnergyTransferSuspended
System under test	CSMS
Description	This state will simulate that the Charging Station is in a state where the energy transfer is suspended by the EV.

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): If State is NOT <i>EnergyTransferStarted</i> then execute Reusable State <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
Notes(s): The tool will wait for <Configured Transaction Duration> seconds	
1. The Test System sends a TransactionEventRequest With triggerReason is <i>ChargingStateChanged</i> transactionInfo.chargingState is <i>SuspendedEV</i>	2. The CSMS responds with a TransactionEventResponse

Tool validations
N/a
Post scenario validations: State is <i>EnergyTransferSuspended</i>

StopAuthorized

State	StopAuthorized
System under test	CSMS
Description	This state will simulate that the Charging Station is in a state where the charging session is authorized to stop.

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): If State is NOT <i>EnergyTransferStarted</i> then execute Reusable State <i>EnergyTransferStarted</i>

Main (Test scenario)	
Charging Station	CSMS
Notes(s): The tool will wait for <Configured Transaction Duration> seconds	
1. The Test System sends a TransactionEventRequest With triggerReason is <i>StopAuthorized</i> eventType is <i>Updated</i>	2. The CSMS responds with a TransactionEventResponse

Tool validations
* Step 2: Message: TransactionEventResponse - idTokenInfo.status must be <i>Accepted</i>
Post scenario validations: State is <i>StopAuthorized</i>

EVConnectedPostSession

State	EVConnectedPostSession
System under test	CSMS
Description	This state will simulate that the Charging Station is in a state where the energy transfer has been stopped and the transaction is NOT authorized to resume energy transfer without re-authorization.

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): If State is NOT <i>StopAuthorized</i> then execute Reusable State <i>StopAuthorized</i>

Main (Test scenario)	
Charging Station	CSMS
1. The Test System sends a TransactionEventRequest With triggerReason is <i>ChargingStateChanged</i> transactionInfo.chargingState is <i>EVConnected</i> eventType is <i>Updated</i>	2. The CSMS responds with a TransactionEventResponse

Tool validations
N/a
Post scenario validations: State is <i>EVConnectedPostSession</i>

EVDisconnected

State	EVDisconnected
System under test	CSMS
Description	This state will simulate that the EV and EVSE of the simulated Charging Station are disconnected, after the charging session is authorized to stop.

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): If State is NOT <i>EVConnectedPostSession</i> then execute Reusable State <i>EVConnectedPostSession</i>

Main (Test scenario)	
Charging Station	CSMS
1. The Test System notifies the CSMS about the status change of the connector. Message: StatusNotificationRequest - connectorStatus is <i>Available</i> Message: NotifyEventRequest - trigger is <i>Delta</i> - actualValue is <i>Available</i> - component.name is <i>Connector</i> - variable.name is <i>AvailabilityState</i>	2. The CSMS responds accordingly.
3. The Test System sends a TransactionEventRequest With triggerReason is <i>EVCommunicationLost</i> transactionInfo.chargingState is <i>Idle</i> transactionInfo.stoppedReason is <i>EVDisconnected</i> eventType is <i>Ended</i>	4. The CSMS responds with a TransactionEventResponse

Tool validations
N/a
Post scenario validations: State is <i>EVDisconnected</i>

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 (Test scenario)

Charging Station	CSMS
Manual Action: <i>Trigger the CSMS to send a GetInstalledCertificateIdsRequest with certificateType _<Specified certificateType></i>	
2. The Test System responds with a GetInstalledCertificateIdsResponse With status is <i>Accepted</i> certificateHashDataChain contains an entry with following values: certificateHashDataChain[0].certificateType is <i><Specified certificateType></i> certificateHashDataChain[0].certificateHashData contains <i><HashData from the configured certificate of the specified certificateType></i>	1. The CSMS sends a GetInstalledCertificateIdsRequest

Tool validations

* Step 1:

Message: **GetInstalledCertificateIdsRequest**- **certificateType** must be *<Specified certificateType>*

Post scenario validations:

Certificate of the specified certificateType is retrieved from the Charging Station.

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 (Test scenario)

Charging Station	CSMS
Manual Action: <i>Trigger the CSMS to send an InstallCertificateRequest with certificateType <Specified certificateType></i>	
2. The Test System responds with a InstallCertificateResponse With status is <i>Accepted</i>	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 scenario validations:

Certificate of the specified certificateType is stored at the Charging Station.

ISO15118SmartCharging

State	ISO15118SmartCharging
System under test	CSMS
Description	A version for the ISO15118 can be specified as argument to this reusable state. This version is exposed as variable <i><iso15118Version></i> and can either be <i>iso15118-2</i> or <i>iso15118-20</i> . If not specified <i>iso15118-2</i> will be assumed.

Before (Preparations)

Configuration State:

N/a

Memory State:

N/a

Reusable State(s):

N/a

Main (Test scenario)

Charging Station	CSMS
-------------------------	-------------

Main (Test scenario)	
<p>1. The Test System sends a NotifyEVChargingNeedsRequest with</p> <p>evseld <Configured evseld></p> <p>maxScheduleTuples 2</p> <p>if <iso15118Version> is iso15118-2:</p> <p>chargingNeeds.requestedEnergyTransfer AC_three_phase</p> <p>chargingNeeds.acChargingParameters.energyAmount 50000</p> <p>chargingNeeds.acChargingParameters.evMinCurrent 0</p> <p>chargingNeeds.acChargingParameters.evMaxCurrent 32</p> <p>chargingNeeds.acChargingParameters.evMaxVoltage 230</p> <p>end if</p> <p>if <iso15118Version> is iso15118-20:</p> <p>chargingNeeds.requestedEnergyTransfer AC_single_phase</p> <p>chargingNeeds.availableEnergyTransfer [AC_single_phase, AC_three_phase, AC_BPT]</p> <p>chargingNeeds.controlMode ScheduledControl</p> <p>chargingNeeds.v2xChargingParameters.maxChargePower 10000</p> <p>chargingNeeds.v2xChargingParameters.maxDischargePower 10000</p> <p>end if</p>	<p>2. The CSMS responds with a NotifyEVChargingNeedsResponse.</p>
<p>4. The Test System responds with a SetChargingProfileResponse with:</p> <p>status Accepted</p>	<p>3. The CSMS sends a SetChargingProfileRequest</p> <p><u>Note(s):</u></p> <p>- If NotifyEVChargingNeedsResponseStatus was Processing, the Test System will wait 60 seconds for the request</p>
<p>5. The Test System sends a NotifyEVChargingScheduleRequest with</p> <p>evseld <COnfigured evseld></p> <p>chargingSchedule <ChargingSchedule provided at step 3></p>	<p>6. The CSMS responds with a NotifyEVChargingScheduleResponse.</p>
<p>7. The Test System sends a TransactionEventRequest with</p> <p>triggerReason <ChargingStateChanged></p> <p>transactionInfo.chargingState <Charging></p>	<p>8. The CSMS responds with a TransactionEventResponse.</p>

Tool validations
<p>* Step 2:</p> <p>Message: NotifyEVChargingNeedsResponse</p> <p>- Status must be Accepted or Processing</p> <p>* Step 3:</p> <p>Message: SetChargingProfileRequest</p> <p>- chargingProfilePurpose must be <TxProfile></p> <p>- transactionId must be <Provided transactionId from before></p> <p>* Step 4:</p> <p>Message: NotifyEVChargingScheduleResponse</p> <p>- status must be <Accepted></p>
N/a

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
<u>Manual Action</u> : Request the CSMS to send a Trigger Message Request with requestedMessage SignChargingStationCertificate	
2. The Test System sends a TriggerMessageResponse with status Accepted	1. The CSMS sends a TriggerMessageRequest With requestedMessage SignChargingStationCertificate
3 The Test System sends a SignCertificateRequest	4. The CSMS responds with a SignCertificateResponse with status Accepted
6. The Test System 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

ISO15118-20 transaction started in V2X AC_BPT energy transfer mode

State	ISO15118-20 transaction started in V2X AC_BPT energy transfer mode
System under test	CSMS
Description	Starts ISO15118-20 transaction and upgrades to V2X AC_BPT energy transfer mode
Exposes	<allowedEnergyTransferModes> <transactionId>

Before (Preparations)
Configuration State: N/a
Memory State: N/a
Reusable State(s): State is ISO15118-20 transaction just started exposing: - <allowedEnergyTransferModes> - <transactionId>

Main (Test scenario)	
Charging Station	CSMS
Agree on energy transfer mode	
<p>1. The Test System sends a NotifyEVChargingNeedsRequest with:</p> <p>evseld = <Configured evseld></p> <p>chargingNeeds.requestedEnergyTransfer = AC_BPT</p> <p>chargingNeeds.availableEnergyTransfer = [AC_single_phase, AC_two_phase, AC_three_phase, AC_BPT, AC_BPT_DER, AC_DER]</p> <p>chargingNeeds.controlMode = ScheduledControl</p> <p>chargingNeeds.v2xChargingParameters.maxChargePower = 1234W</p> <p>chargingNeeds.v2xChargingParameters.maxDischargePower = 0W</p>	<p>2. The CSMS responds with a NotifyEVChargingNeedsResponse</p>

Tool validations
* Step 2: Message: NotifyEVChargingNeedsResponse - status must be Accepted or Processing
Post scenario validations: N/a

ISO15118-20 transaction just started

State	ISO15118-20 transaction just started
System under test	CSMS
Description	Starts ISO15118-20 transaction
Exposes	<allowedEnergyTransferModes> <transactionId>
Prerequisite(s)	Test System reported support for V2XChargingCtrlr.V2XEnabled for configured EVSEs with value <i>true</i>

Before (Preparations)

Configuration State:

N/a

Memory State:

N/a

Reusable State(s):

State is *EVConnectedPreSession*

Main (Test scenario)

Charging Station	CSMS
1. The Test System sends an AuthorizeRequest with idToken.type <i>eMAID</i> idToken.type <Configured ISO15118 idToken> iso15118CertificateHashData <hashData's from MO certificate chain>	2. The CSMS responds with an AuthorizeResponse - allowedEnergyTransfer [<csms_allowedEnergyTransferModes>]
3. The Test System sends a TransactionEventRequest with eventType <i>Started</i> transactionInfo.transactionId <transactionId> idToken.additionalInfo.additionalIdToken <Configured EVCCID> idToken.additionalInfo.additionalInfo.type <i>EVCCID</i>	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